# Wing IDE Benutzerhandbuch
# Wing IDE Professional

Wingware
www.wingware.com

Version 2.1.0
April 17, 2006

## Inhalt

Einleitung

- 1.1. Produktebenen
- 1.2. Lizenzen
- 1.3. Unterstützte Plattformen
- 1.4. Unterstützte Python-Versionen
- 1.5. Technischer Support
- 1.6. Grundvoraussetzungen für die Installation
- 1.7. Installation
- 1.8. Ausführung des IDEs
- 1.9. Installation Ihrer Lizenz
- 1.10. Verzeichnis der Benutzereinstellungen
- 1.11. Aufrüsten (Upgrade)
  - 1.11.1. Ein gescheitertes Upgrade beheben
- 1.12. Erweiterte Installation
  - 1.12.1. Installation zusätzlicher Dokumentation
  - 1.12.2. Installationshinweise für Linux
  - 1.12.3. Source-Code-Installation
- 1.13. Wing IDE entfernen
- 1.14. Verwendung der Befehlszeile
- 1.15. Fehlerbehebung
  - 1.15.1. Fehlerbehebung für Startfehler
  - 1.15.2. Probleme in Microsoft Windows
  - 1.15.3. Fehlerbehebung für Debug-Fehler
    - 1.15.3.1. Fehler beim Starten des Debug-Prozesses
    - 1.15.3.2. Zusätzliche Exceptions im Debugger
    - 1.15.3.3. Fehler beim Stoppen an Exceptions
    - 1.15.3.4. Debugger stoppt nicht an Haltepunkten oder zeigt Source-Code nicht an
  - 1.15.4. Diagnoseausgabe erhalten
  - 1.15.5. Wing IDE beschleunigen
  - 1.15.6. Fehlerbehebung öffnungs-Fehler der Dateinamen mit Leerzeichen

Anpassung

- 2.1. Optionen der Benutzeroberfläche

2.1.1. Fensteraufteilungen
2.1.2. Layout der Benutzeroberfläche
2.1.3. Änderung der Textanzeige
2.1.4. Einstellung des insgesamten Anzeigethemas
2.2. Einstellungen
2.2.1. Ebenen der Einstellungsdatei
2.2.2. Format der Einstellungsdatei
2.3. Editor-Individualitäten
2.4. Tastaturbefehle
2.4.1. Tastennamen
2.5. Datei-Sets
Projektmanager
3.1. Ein Projekt erstellen
3.2. Dateien und Pakete entfernen
3.3. Das Projekt speichern
3.4. Die Ansicht sortieren
3.5. Tastaturnavigation
3.6. Gemeinsame Nutzung von Projekten
3.7. Projektweite Eigenschaften
3.8. Pro-Datei Eigenschaften
3.9. Dateiinformationen anzeigen
3.10. Navigation zu Dateien
Source-Code-Editor
4.1. Syntax-Farbmarkierung
4.2. Rechtsklick-Menü des Editors
4.3. Source-Code-Navigation
4.4. Dateistatus und nur lesbare Dateien
4.5. Vorübergehende vs. nicht vorübergehende Editoren
4.6. Strukturelles Falten
4.7. Klammernsuche
4.8. Einrückung
4.8.1. Automatisch Einrücken
4.8.2. Die Tab-Taste

4.8.3. Einrückung überprüfen
4.8.4. Blockeinrückung ändern
4.8.5. Einrückungsmanager
4.9. Auto-Vervollständigung
4.10. Source-Assistent
4.11. Automatisch speichern
4.12. Hinweise zu Kopieren/Einfügen
4.13. Geänderte Dateien automatisch Neuladen
4.14. Suchen/Ersetzen
4.14.1. Schnellsuche mit der Werkzeugleiste
4.14.2. Tastaturgesteurte Mini-Suche/Ersetzen
4.14.3. Suchen/Ersetzen-Werkzeug
4.14.3.1. Modi und Bereich für Suchen/Ersetzen
4.14.3.2. Optionen für Suchen/Ersetzen
4.14.3.3. Suchergebnisse ersetzen
4.15. User-defined Bookmarks
4.16. Templating (Code Snippets)
Overview
Syntax
Indentation and Line Endings
Cursor Placement
Reloading
Commands
User Interface
4.17. Using Revision Control with Wing
Installing CVS
Installing Subversion
Using SSH Repositories
Subversion with SSH
Subversion with http/https or file URLs
Subversion without Authentication Cache
Using CVS with SSH
Using CVS with pserver
Notes on the Implementation
4.18. Tastaturmakros
4.18.1. Beispiel eines Makros

4.19. Source-Code-Analyse
4.19.1. Analyse-Cache
Source-Code-Browser
5.1. Wahlmöglichkeiten für die Anzeige
5.1.1. Nach Modul anzeigen
5.1.2. Klassenhierarchie anzeigen
5.1.3. Alle Klassen anzeigen
5.2. Anzeigefilter
5.2.1. Bereich und Source-Code filtern
5.2.2. Konstrukttyp filtern
5.3. Die Browser-Anzeige sortieren
5.4. Navigation der Ansichten
5.5. Tastaturnavigation des Browsers
Debugger
6.1. Schnellstart
6.2. Bestimmung des Debug-Startpunktes
6.3. Debug-Eigenschaften
6.4. Haltepunkte setzen
6.5. Debuggen starten
6.6. Debugger-Status
6.7. Ablaufsteuerung
6.8. Stack anzeigen
6.9. Debug-Daten anzeigen
6.9.1. Ansicht der Stack-Daten
6.9.1.1. Optionen des Popup-Menüs
6.9.1.2. Anzeige von Werten filtern
6.9.2. Werte verfolgen
6.9.3. Ausdrücke bewerten
6.9.4. Probleme bei der Behandlung von Werten
6.10. Interaktiver Debug-Test
6.11. Interaktive Python-Shell
6.12. Exceptions verwalten
6.13. Debug-Prozess-I/O

6.13.1. Externe I/O-Konsolen
6.13.2. Multiplex-Betrieb des Debug-Prozess-I/Os deaktivieren
6.14. Anhängen und Abtrennen
6.14.1. Zugriffskontrolle
6.14.2. Abtrennen
6.14.3. Anhängen
6.14.4. Fremde Prozesse identifizieren
6.14.5. Beschränkungen
6.15. Extern gestarteten Code debuggen
6.15.1. Import des Debuggers
6.15.2. Konfiguration des Debug-Servers
6.15.3. Remote-Debuggen
6.15.4. Abbildung der Dateiposition
6.15.4.1. Beispiele für die Abbildung der Dateiposition
6.15.5. Beispiel für das Remote-Debuggen
6.15.6. Debugger-API
6.16. Ohne Debuggen ausführen
6.17. Beschränkungen des Debuggers
Scripting and Extending Wing IDE
7.1. Scripting Example
7.2. Getting Started
Naming Scripts
Reloading Scripts
Overriding Internal Commands
7.3. Script Syntax
Script Attributes
ArgInfo
Commonly Used Types
Commonly Used Formlets
Magic Default Argument Values
GUI Contexts
Top-level Attributes
Importing Other Modules
Internationalization and Localization
7.4. Scripting API

7.5. Advanced Scripting
Example
How Script Reloading Works
7.6. Known Scripting Issues
Referenz der Einstellungen
Benutzeroberfläche
Dateien
Editor
Debugger
Source-Analyse
IDE Extension Scripting
Interne Einstellungen
Haupteinstellungen
Einstellungen der Benutzeroberfläche
Einstellungen des Editors
Einstellungen des Projektmanagers
Einstellungen des Debuggers
Einstellungen der Source-Analyse
Einstellungen des Source-Browsers
Befehlsreferenz
Top Level Commands
Dock Window Commands
Document Viewer Commands
Editor Browse Mode Commands
Editor Insert Mode Commands
Editor Non Modal Commands
Editor Panel Commands
Editor Replace Mode Commands
Editor Split Commands
Editor Visual Mode Commands
Global Documentation Commands
Toolbar Search Commands

- Window Commands
- Wing Tips Commands
- Active Editor Commands
- General Editor Commands
- Project Manager Commands
- Project View Commands
- Debugger Commands
- Debugger Watch Commands

Lizenzinformationen

- 10.1. Wing IDE Software-Lizenz
- 10.2. Open Source Lizenzinformationen


---




```
Wingware
P.O. Box 1937
Brookline, MA 02446
United States of America
```

# Einleitung

Vielen Dank, dass Sie sich für Wing IDE von Wingware entschieden haben! Das Handbuch wird Ihnen beim Starten helfen und dient als Referenz für das gesamte Funktionsset des Produkts.

Das Handbuch ist nach Hauptfunktionsbereichen von Wing IDE gegliedert, was den Projektmanager, Source-Code-Editor, Source-Browser (nur in Wing IDE Professional) und Debugger beinhaltet. Mehrere Anhänge dokumentieren das gesamte Befehlsset, stellen Hinweise zu Ressourcen und Tipps für Wing- und Python-Nutzer bereit und führen die volle Software-Lizenz auf.

Der Rest dieses Kapitels beschreibt, wie Sie Wing IDE installieren und starten. Wenn Sie Handbücher nicht gern lesen, sollten Sie in der Lage sein, das Produkt zum Laufen zu bringen, indem Sie nur dieses Kapitel lesen.

**Schlüsselkonzepte**
In dem Handbuch sind Schlüsselkonzepte, wichtige Hinweise und nicht offensichtliche Funktionen genauso wie dieser Paragraph hervorgehoben. Wenn Sie den Text nur überfliegen, dann suchen Sie nach diesen Markierungen.

Beachten Sie, dass der gesamte Inhalt des Handbuchs auch innerhalb von Wing IDE durch den Hilfemanager verfügbar ist.

## 1.1. Produktebenen

Dieses Handbuch ist für das Produktlevel Wing IDE Professional aus der Wing IDE Produktlinie, welche zur Zeit Wing IDE Personal und Wing IDE Professional umfasst.

Wing IDE Professional ist das voll funktionsfähige Wing IDE Produkt und kann sowohl für den kommerziellen als auch nicht-kommerziellen Gebrauch lizenziert werden. Wing IDE Personal ist ausschließlich für den nicht-kommerziellen Gebrauch bestimmt und

enthält nur einen Teil der Funktionen, die in Wing IDE Professional zur Verfügung stehen.

Wing IDE Professional und Wing IDE Personal sind unabhängige Produkte, die gleichzeitig auf Ihrem System installiert sein können, ohne miteinander in Konflikt zu geraten.

Eine Liste der Funktionen, die in Wing IDE Personal nicht verfügbar sind, finden Sie auf http://wingware.com/wingide/features.

## 1.2. Lizenzen

Die Lizenzierung für Wing IDE erfolgt pro Entwickler und sie erfordert eine separate Lizenz für jedes Betriebssystem, dass von dem Entwickler verwendet wird. Lizenzen, die für mehrere Benutzer erworben wurden, erlauben, dass bis zur erworbenen Anzahl Nutzer Wing gleichzeitig auf dem lizenzierten Betriebssystem ausführen können. Lizenzpakete für mehrere Betriebssysteme sind in unserem Online-Shop zu Rabattpreisen erhältlich.

Den vollständigen Text unserer Lizenz finden Sie unter **Software-Lizenz**.

**Lizenzaktivierung**

Wing IDE erfordert die Aktivierung einer Probe- oder dauerhaften Lizenz, wenn es länger als 10 Minuten ausgeführt werden soll. Dieses System ist so entworfen, dass es die Belastung für rechtmäßige Nutzer minimiert, aber auch, dass es die leichtfertige gemeinsame Nutzung von Lizenzen ausschaltet, die unsere Fähigkeit, die weitere Entwicklung von Wing IDE zu unterstützen, gefährdet.

Eine Aktivierung bindet die Lizenz an die Maschine; dies geschieht durch eine Reihe von Überprüfungen der Hardware, die mit dem System verbunden ist. Diese Informationen werden niemals über das Internet übertragen, sondern stattdesssen wird ein SHA-Bash von einigen Werten hin und her geschickt, so dass die Maschine identifizierbar ist, ohne das wir spezifische Dinge über sie wissen.

Die Metrik der Maschinenidentität, die für die Aktivierung verwendet wird, ist so erstellt, dass das Ersetzen von Hardware-Teilen Ihrer Maschine oder das Aufrüsten der Maschine normalerweise keine weitere Aktivierung erfordert. Aus dem gleichen Grund erhöht die mehrfache Aktivierung auf der gleichen Maschine (zum Beispiel wenn die Aktivierungsdatei verloren gegangen ist) Ihre Aktivierungsanzahl nicht.

Lizenzen werden standardmäßig mit drei Aktivierungen geliefert. Zusätzliche Aktivierungen können selbständig mit dem Lizenzmanager erworben werden oder indem eine E-Mail an sales at wingware.com geschickt wird. Als Absicherung für Notfälle, in denen

wir nicht kontaktiert werden können und Sie keine Aktivierung haben, kann Wing IDE für jeweils 10 Minuten ohne Lizenz ausgeführt werden.

Siehe **Installation Ihrer Lizenz** für zusätzliche Informationen über das Erlangen und Aktivieren von Lizenzen.

## 1.3. Unterstützte Plattformen

Diese Version von Wing IDE ist für Microsoft Windows, Linux und Mac OS X erhältlich. Außerdem steht es für einige andere Betriebssysteme zur Verfügung, für die Builds von anderen Nutzern bereitgestellt wurden oder bei denen Kunden bereit sind, das Produkt vom Source-Code zu kompilieren.

**Microsoft Windows**

Wing IDE unterstützt Windows 98 mit IE5+ (*), ME mit IE5+ (*), NT4 mit IE5+ (*), 2K, XP sowie 2003 Server. Windows 95 wird nicht unterstützt.

(*) In Windows 98 können ME und NT4 Installationen mit diesem Patch von Microsoft höchstwahrscheinlich verwendet werden, anstatt zu IE5 aufzurüsten.

**Linux/Intel**

Wing IDE läuft auf Linux-Versionen mit glibc2.2 oder höher (alles, das ungefähr 3 Jahre alt oder neuer ist sollte funktionieren; zum Beispiel RedHat 7.1+, Mandrake 8.0+, SuSe 7.1+ und Debian 3.0+).

In Suse müssen Sie die gmp- und python-Pakete installieren oder Python vom Source-Code installieren, da Python hier standardmäßig nicht installiert ist.

In Debian können Sie das Wing IDE RPM-Paket in ein Debian-freundliches Paket umwandeln, indem Sie das `alien` Modul verwenden. Installieren Sie das `alien`-Paket und führen dann `alien -d wingide-*.i386.rpm` aus, gefolgt von `dpkg -i wingide-*.deb`. Alternativ können Sie den Wing IDE tar-Datei-Installierer verwenden.

**Mac OS X**

Wing IDE läuft auf Mac OS X 10.1+. Wing IDE für OS X erfordert außerdem einen X11 Server und Fenstermanager. Siehe **OS X Schnellstart-Anleitung** für Einzelheiten.

Für Mac OS X wird nur Python 2.2 oder höher unterstützt. Version 10.3 oder höher von OS X werden jedoch mit einer bereits installierten Standardversion von Python geliefert.

**Andere Plattformen**

Kunden können Wing IDE vom Source-Code kompilieren, wenn Sie es auf anderen Betriebssystemen (wie Linux PPC, Free BSD oder Solaris) verwenden möchten. Dies erfordert die Unterzeichnung einer Geheimhaltungsvereinbarung.

Einige Builds für Wing IDE, die von anderen Nutzern bereitgestellt wurden, sind auch für andere Betriebssysteme verfügbar.

## 1.4. Unterstützte Python-Versionen

Vor der Installation von Wing, müssen Sie zunächst Python 1.5.2, Python 2.0, Python 2.1, Python 2.2, Python 2.3 oder Python 2.4 herunterladen und installieren, wenn Sie nicht bereits eine dieser Versionen auf Ihrer Maschine haben.

*In Windows* muss Python mit einem der Installer von python.org installiert werden (oder es muss vom Source-Code erstellt werden, wenn gewünscht).

*In Linux* kommen die meisten Distributionen mit Python. Die Installation von Python ist normalerweise nur in Suse, RedHat 6.0 oder einer speziell angepassten Linux-Installation erforderlich.

*In Suse Linux* können Sie die gmp- und Python-Pakete installieren, die mit Ihrer Distribution kommen oder Sie installieren Material, das von den oben genannten Links bereitgestellt wird.

*In RedHat 6.0* müssen Sie Python 1.5.2 oder höher installieren und dies zur Ausführung Ihres Debug-Programms und dem tar-Datei-Installierer (wenn nicht von RPM installiert) verwenden. Wing funktioniert mit der standardmäßigen 1.5.1 Installation, die mit RedHat 6.0 kommt, nicht.

*Auf Mac OS X* unterstützt Wing IDE nur Python 2.2 oder höher.

## 1.5. Technischer Support

Wenn Sie bei der Installation oder Nutzung von Wing IDE Probleme haben, reichen Sie uns bitte einen Fehlerbericht oder Feedback ein. Verwenden Sie dafür die Einträge `Fehlerbericht einreichen` oder `Feedback einreichen` in Wing IDE's `Hilfemenü`.

Sie können auch Wingware's Technischen Support per E-Mail unter support at wingware.com kontaktieren oder online unsere Support-Website http://wingware.com/support besuchen.

Fehlerberichte können auch per E-Mail an bugs at wingware.com gesendet werden. Bitte

geben Sie in jedem Bericht Ihr Betriebssystem, die Produktversionsnummer und Einzelheiten des Problems an.

Wenn Sie einen Fehlerbericht per E-Mail einreichen, lesen Sie bitte auch den Abschnitt **Diagnoseausgabe erhalten**, um zusätzliche Informationen darüber zu bekommen, wie Sie ein Protokoll von Wing IDE und der Internals des Debug-Prozesses erfassen. Wenn möglich sollten diese Daten bei Fehlerberichten per E-Mail enthalten sein.

## 1.6. Grundvoraussetzungen für die Installation

Zur Ausführung von Wing IDE müssen Sie die folgenden Dinge erwerben und installieren, wenn Sie nicht bereits auf Ihrem System vorhanden sind:

**Grundvoraussetzungen für alle Plattformen:**

- Version von Wing IDE, entweder heruntergeladen oder von CD
- Eine **unterstützte Python-Version**
- Eine funktionierende TCP/IP Netzwerk-Konfiguration
- Internet Explorer, Netscape oder anderer Web-Browser (optional)
- Adobe Acrobat Reader 4.0.5 oder höher (optional)

**Zusätzliche Grundvoraussetzungen für Mac OS X:**

- Ein X Window-Server, wie Apple X11 für OS X oder XDarwin
- Ein Fenstermanager. Apple's Server beinhaltet einen; andere Optionen sind Window Maker und OroborOSX

## 1.7. Installation

Versichern Sie sich vor der Installation von Wing IDE, dass die **notwendigen Grundvoraussetzungen** installiert sind. Wenn Sie eine vorherige Version aufrüsten, lesen Sie zuerst den Abschnitt **Aufrüsten**. Eine schnelle Einleitung zu den Funktionen von Wing IDE ist in der **Wing IDE Schnellstart-Anleitung** zu finden.

**Hinweis:** Auf allen Plattformen wird der Installationsort von Wing IDE als `WINGHOME` bezeichnet.

### Windows 98se, NT 4, Windows 2000 und Windows XP

Installieren Sie Wing IDE, indem Sie die heruntergeladene Executable ausführen. Wing's Dateien werden standardmäßig in `C:\Programme\Wing IDE` installiert, aber dieser Speicherort kann während der Installation verändert werden. Wing wird außerdem das **Verzeichnis der Benutzereinstellungen** an der für Ihre Windows-Version entsprechenden Stelle anlegen. Es wird dazu verwendet, Einstellungen und andere Einrichtungen zu speichern.

### Linux

*Um das RPM zu installieren*, müssen Sie zu Root wechseln und `rpm -i wingide-2.1.0-b2.i386.rpm` eintippen. Das IDE wird in `/usr/lib/wingide2.1` installiert mit Executables in `/usr/bin`. Die Wing IDE Executable ist `/usr/bin/wing2.1`.

*Um die tar-Datei zu installieren*, müssen Sie `tar -zxvf wingide-2.1.0-b2-i386-linux.tar.gz` eintippen, um aus dem tar-Archiv zu extrahieren. Gehen Sie dann mit `cd` zum neu erstellten Verzeichnis `wingide-2.1.0-b2-i386-linux`, tippen `./wing-install.py` ein und beantworten die Fragen, um zu bestimmen, wo die Programmdateien gespeichert werden sollen.

Die Wing IDE Executable heißt `wing2.1`. Sie werden Ihre PATH-Umgebung ändern müssen, wenn Sie die Executable in einem Verzeichnis gespeichert haben, das nicht bereits in Ihrem Pfad ist. Sie werden auch eine neue Shell öffnen müssen oder anfordern, dass Ihre Shell die Festplatte nach Executables neu durchsucht. (zum Beispiel mit `rehash` in tsch).

Wing wird das **Verzeichnis der Benutzereinstellungen** in `~/.wingide2` anlegen. Es wird verwendet, um Einstellungen und andere Einrichtungen zu speichern.

Weitere Informationen finden Sie in **Linux Installationsdetails**

### Mac OS X

Auf Mac OS X erfordert Wing IDE, dass Sie zuerst (a) Python 2.2 oder höher (frühere Versionen werden nicht funktionieren) und (b) einen X Server und Fenstermanager installieren. Einzelheiten zur Installation und Ausführung auf OS X finden Sie unter **Wing IDE für OS X**.

## 1.8. Ausführung des IDEs

Für eine schnelle Einführung zu Wing's Funktionen lesen Sie bitte die **Wing IDE Schnellstart-Anleitung**. Für einen sanfteren, ausführlicheren Start lesen Sie bitte das **Wing IDE Tutorial**.

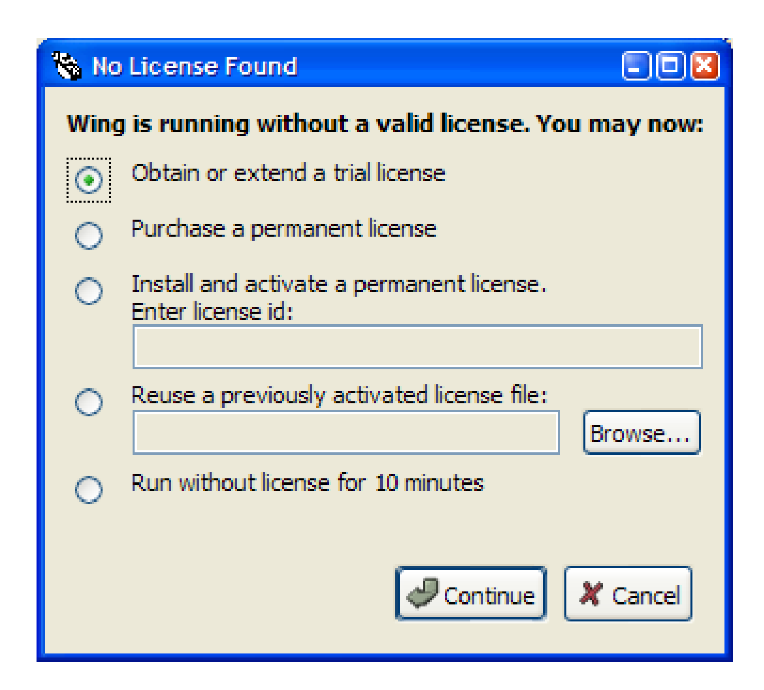
**In Windows** starten Sie Wing IDE aus der Programmgruppe des Startmenüs. Sie können Wing auch von der Befehlszeile mit `wing` (in `WINGHOME` gelegen) starten.

**In Linux/Unix** führen Sie einfach `wing2.1` aus (in `WINGHOME` gelegen).

**In Mac OS X** starten Sie zuerst Ihren X Windows-Server und Fenstermanager. Wenn das erledigt ist, starten Sie Wing IDE mit einem Doppelklick auf den Anwendungsordner. Wenn Sie Wing von der Befehlszeile duch Verwendung von `Contents/MacOS/wing` im Wing IDE Anwendungsordner starten, dann müssen sie Ihre `DISPLAY` Umgebungsvariable setzen.

## 1.9. Installation Ihrer Lizenz

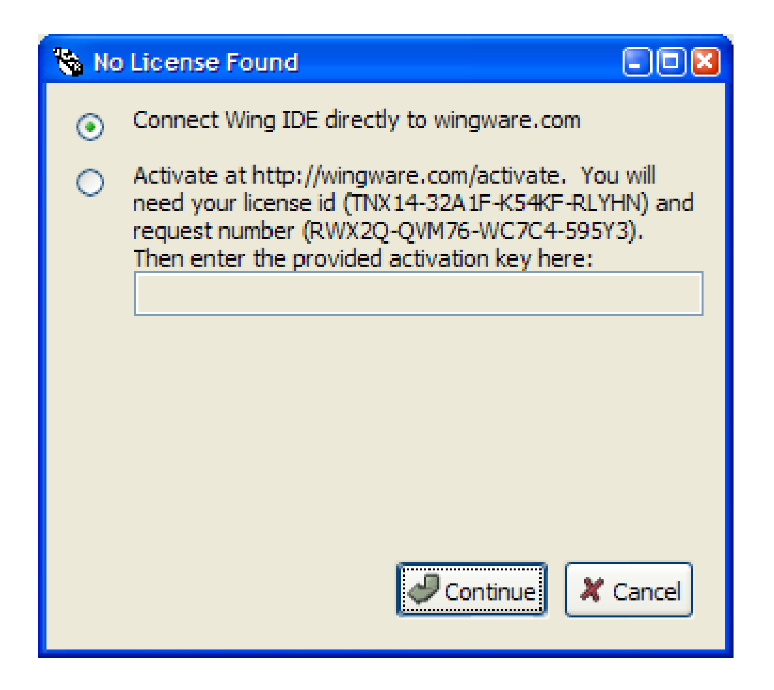
Wing IDE erfordert eine Probe- oder dauerhafte Lizenz, wenn Sie es länger als 10 Minuten ausführen wollen. Die Lizenz muss aktiviert werden (siehe Abschnitt **Lizenzen** für allgemeine Informationen zur Aktivierung). Wenn Wing IDE das erste Mal gestartet wird, können Sie entweder eine Probelizenz erhalten, eine dauerhafte Lizenz erwerben, eine dauerhafte Lizenz installieren und aktivieren, eine bereits aktivierte Lizenz verwenden oder Wing IDE 10 Minuten lang ohne Lizenz nutzen:




### Probelizenzen

Probelizenzen ermöglichen die Bewertung von Wing IDE für 10 Tage mit der Option, die Bewertungszeit um weitere 10 Tage zu verlängern. Der einfachste Weg, eine Probelizenz

zu erwerben, besteht darin, Wing IDE aufzufordern, direkt zu TCP Port 80 (http) auf `wingware.com` zu verbinden. In diesem Fall versucht Wing zu verbinden und erhält alle Informationen, die es benötigt, um 10 Tage lang zu laufen. Nachdem die Probelizenz erworben wurde, wird Wing nicht mehr versuchen, zu `wingware.com` (oder anderen Websites) zu verbinden, es sei denn, Sie reichen Feedback oder einen Fehlerbericht über das Menü `Hilfe` ein.




Wenn Sie Wing IDE nicht direkt zu `wingware.com` verbinden können oder wollen, können Sie auch zu http://wingware.com/activate gehen und die Lizenz-ID und Aktivierungsabfragenummer eingeben. Nach der Eingabe dieser Informationen erhalten Sie einen Aktivierungsschlüssel, den Sie in Wing's Dialogbox eingeben können, um die Aktivierung abzuschließen. Hierbei passiert genau der gleiche Informationsaustausch, wie bei der Variante, bei der Wing IDE direkt zu `wingware.com` verbindet, um eine Probelizenz zu erhalten.

Bitte kontaktieren Sie uns per E-Mail unter sales at wingware.com, wenn Probleme auftreten oder Sie zusätzliche Zeit zur Bewertung von Wing IDE benötigen.

### Dauerhafte Lizenzen

Dauerhafte Lizenzen und Upgrades können in unserem Online-Shop unter http://wingware.com/store erworben werden. Dauerhafte Lizenzen beinhalten freie Upgrades in der 2.* Versionsserie. Lizenzen für Wing IDE Professional umfassen außerdem Zugriff auf den Source-Code des Produktss über http://wingware.com/downloads (erfordert Geheimhaltungsvereinbarung).

**Aktivierung auf gemeinsam genutzten Laufwerken**

Bei der Installation von Wing auf einem gemeinsam genutzten Laufwerk (zum Beispiel einem USB Keydrive oder einem Datei-Server) kann auf das **Verzeichnis der Benutzereinstellungen**, in welchem die Lizenzaktivierung gespeichert ist, von mehreren Computern zugegriffen werden.

In diesem Fall muss Wing auf jedem Computer einmal aktiviert werden. Die daraus resultierenden extra Aktivierungen werden als `license.act1`, `license.act2` und so weiter gespeichert. Je nachdem, wo Wing ausgeführt wird, wird es automatisch die entsprechende Aktivierung auswählen.

**Erhalt zusätzlicher Aktivierungen**

Wenn Sie keine Aktivierungen mehr haben, können Sie entweder den Lizenzmanager verwenden oder eine E-Mail an sales at wingware.com schicken, um zusätzliche Aktivierungen für rechtmäßig erworbene Lizenzen zu erhalten.

**Wiederverwendung einer bestehenden Aktivierung**

Wenn Sie eine Maschine aufrüsten und beabsichtigen, Wing IDE auf dieser Maschine weiter zu verwenden, müssen Sie eine Sicherungskopie der Datei `license.act` in Ihrem **Verzeichnis der Benutzereinstellungen** erstellen. Nachdem die Maschine aufgerüstet und Wing IDE neu installiert ist, können Sie die Option `Eine bereits aktivierte Lizenzdatei wiederverwenden` auswählen. Dies aktiviert Wing sofort, ohne zu `wingware.com` zu verbinden. Diese Möglichkeit funktioniert allerdings nicht, wenn Sie Ihre Hardware komplett ersetzt und gleichzeitig andere Attribute Ihre Maschine geändert haben.

In Fällen, in denen eine Lizenz neu aktiviert werden muss, weil die Lizenzaktivierungsdatei verloren gegangen ist, wird sich Ihre Aktivierungsanzahl nur erhöhen, wenn sich die Hardware geändert hat.

## 1.10. Verzeichnis der Benutzereinstellungen

Wing wird bei der ersten Ausführung automatisch Ihr `Verzeichnis der Benutzereinstellungen` anlegen. Dieses Verzeichnis speichert Ihre Lizenz, Ihre Einstellungen, automatisch gespeicherte Dateien, den Source-Analyse-Cache, zuletzt verwendete Listen und andere, von Wing intern verwendete Dateien. Wenn dieses Verzeichnis nicht erstellt werden kann, wird Wing beenden.

Das Einstellungsverzeichnis wird an einem Ort erstellt, der für Ihr Betriebssystem geeignet ist. Der Speicherort ist in der **Über Wing IDE Box**, die über das Menü `Hilfe` erreicht werden kann, als Ihr `Einstellungsverzeichnis` gelistet.

Dies sind die von Wing verwendeten Speicherorte:

- **Linux/Unix** -- `~/.wingide2` (ein Unterverzeichnis Ihres Home-Verzeichnisses)
- **Windows** -- In `Wing IDE 2` innerhalb des `Application Data` Ordners. Der Speicherort variiert in Abhängigkeit von der Windows-Version. Die unten aufgelisteten Verzeichnisse sind die Voreinstellungen für die englischsprachige Version von Windows und ein Systemlaufwerk `c:`. Sehen Sie in den Informationen nach, die in der Dialogbox `Über Wing IDE` aufgelistet sind, um das tatsächlich verwendete Verzeichnis zu bestimmen.

  *Windows 98 und ME* -- c:\Windows\Application Data

  *Windows NT* -- c:\WINNT\Profiles\${Benutzername}\Application Data

  *Windows 2000 und XP* -- c:\Documents and Settings\${Benutzername}\Application Data

## 1.11. Aufrüsten (Upgrade)

Wenn Sie Wing aufrüsten und vorher Patch-Dateien installiert haben, lesen Sie diese **zusätzlichen Informationen**, um Probleme während des Upgrades zu vermeiden.

Wenn Sie innerhalb der gleichen Unterversionsnummer von Wing IDE aufrüsten (zum Beispiel von 1.1.8 auf 1.1.10), wird dies Ihre vorherige Installation ersetzen. Sobald Sie aufgerüstet haben, sollten Ihre vorherigen Einstellungen noch vorhanden sein und Sie sollten in der Lage sein, Wing sofort zu starten.

Wenn Sie über Haupt-Releases aufrüsten (zum Beispiel von 2.0 auf 2.1), wird dies eine neue Version installieren, die neben der alten Version existiert.

Beachten Sie, dass die Einstellungen in Wing IDE 2.x vollkommen separat von den Werten sind, die Sie in allen früheren Wing IDE 1.1 Installationen bestimmt haben. Wing 2.x wird die anfänglichen Werte auf allen in Ihrer 1.1 Installation gefundenen Werten basieren, aber dies wird nur das erste Mal gemacht, wenn Sie Wing IDE 2.x starten.

Um ein Upgrade zu installieren, folgen Sie den Schritten, die im Abschnitt **Installation** beschrieben sind.

### 1.11.1. Ein gescheitertes Upgrade beheben

In seltenen Fällen, wenn Sie innerhalb von Unterversionen aufrüsten (zum Beispiel von 2.0 zu 2.0.1), kann das Aufrüsten daran scheitern, alte Dateien zu überschreiben, was zu

zufälligen oder bizarren Verhalten oder Abstürzen führt. Sie beheben dieses Problem, indem Sie Wing vollständig deinstallieren und verbleibende Dateien manuell entfernen, bevor Sie das Upgrade noch einmal installieren.

**Windows**

Für die Deinstallation in Window verwenden Sie die Schaltfläche `Programme Hinzufügen/Entfernen`, um Wing IDE zu deinstallieren. Gehen Sie dann in das Verzeichnis, in welchem Wing platziert war und entfernen manuell alle verbleibenden Ordner und Dateien.

**Linux RPM**

Wenn Sie Wing IDE für Linux von RPM installiert haben, erteilen Sie den Befehl `rpm -e wingide`. Gehen Sie dann in `/usr/lib/wingide` und entfernen manuell alle verbleibenden Dateien und Verzeichnisse.

**Linux Tar**

Wenn Sie Wing IDE für Linux aus der tar-Distribution installiert haben, suchen Sie Ihr Wing Installationsverzeichnis und führen Sie das dort platzierte Skript `wing-uninstall` aus. Ist das erledigt, entfernen Sie manuell alle verbleibenden Dateien und Verzeichnisse.

**Mac OS X**

Auf Mac OS X entpacken Sie einfach ein Archiv, um die Installation zu bilden. Probleme können auftreten, wenn Sie dies über eine bestehende Installation machen. Um das zu vermeiden, entpacken Sie an einer anderen Stelle.

## 1.12. Erweiterte Installation

Dieser Abschnitt beschreibt Installationsoptionen für den fortgeschrittenen Nutzer.

### 1.12.1. Installation zusätzlicher Dokumentation

Wenn Sie Linux/Unix verwenden, ist das Python-Handbuch in den meisten Installationen nicht enthalten. Sie können allerdings lokale Kopien dieser Seiten herunterladen und installieren.

Platzieren Sie die höchste Ebene des HTML-formatierten Python-Handbuches (dort, wo `index.html` zu finden ist) in `python-manual/#.#` in Ihrer Wing IDE Installation. Ersetzen Sie `#.#` mit der Haupt- und Unterversion des entsprechenden Python-Interpreters. (Verwenden Sie beispielsweise für das Python 2.3.x Handbuch `python-manual/2.3`.)

Wenn dies vorgenommen wurde, wird Wing die Kopie auf dem lokalen Laufwerk nutzen, anstatt ins Internet zu gehen, wenn das Python-Handbuch aus dem Hilfemenü ausgewählt wird.

### 1.12.2. Installationshinweise für Linux

In Linux kann Wing von RPM oder vom tar-Archiv installiert werden. Verwenden Sie die letztere Variante, wenn Sie auf Ihrer Machine keinen Root-Zugang haben oder Wing irgendwo anders als `/usr/lib/wingide` installieren möchten.

**Installation von RPM:**

Auf RPM-basierten Systemen, wie RedHat und Mandrake, kann Wing von einem RPM-Paket installiert werden. Um es zu installieren, führen Sie `rpm -i wingide-2.1.0-b2.i386.rpm` als Root aus oder verwenden Sie Ihr Lieblingsverwaltungstool für RPM, um das RPM zu installieren. Die meisten Dateien für Wing werden im Verzeichnis `/usr/lib/wingide` gespeichert und ein Link für den `wing2.1` Befehl ist im Verzeichnis `/usr/bin` platziert.

Der Installationsort von Wing wird als `WINGHOME` bezeichnet. Wenn Sie von RPM installiert haben, wird es immer `/usr/lib/wingide` sein.

**Installation vom Tar-Archiv:**

Wing kann auch vom tar-Archiv installiert werden. Dies kann für Systeme verwendet werden, die kein RPM nutzen oder wenn Sie Wing in einem anderen Verzeichnis als `/usr/lib/wingide` installieren möchten. Das Entpacken dieses Archivs mit `tar -zxvf wingide-2.1.0-b2-i386-linux.tar.gz` wird ein `wingide-2.1.0-b2-i386-linux` Verzeichnis erstellen, welches das `wing-install.py` Skript und eine `binary-package.tar` Datei enthält.

Die Ausführung des `wing-install.py` Skripts wird nach einem Speicherort für die Installation der Support-Dateien für Wing (`WINGHOME`) und nach einem Speicherort, an dem ein symbolischer Link zu `wing2.1` erstellt wird, verlangen. Diese Speicherorte sind auf `/usr/local/lib/wingide` beziehungsweise `/usr/local/bin` voreingestellt. Das Installationsprogramm muss Lese-/Schreibzugriff auf beide Verzeichnisse haben und alle Nutzer, die Wing ausführen, müssen auf beide Verzeichnisse Lesezugriff haben.

Der Installationsort von Wing wird als `WINGHOME` bezeichnet. Wenn Sie von tar installiert haben, wird das der Speicherort sein, den Sie auswählen, wenn Sie den Installer ausführen.

**Installation auf Debian Linux:**

Sie können das Linux-RPM in ein Debian-freundliches Paket umwandeln, indem Sie das `alien` Modul verwenden. So funktioniert's:

1) Das RPM-Paket herunterladen.
2) Das alien-Paket installieren, welches Teil der Debian-Paket-Kollektion ist. Verwenden Sie apt-get, kpackage, aptitude oder einen anderen Paketmanager, um es zu finden und zu installieren.
3) Wechseln Sie zu dem Verzeichnis, in dem das Wing IDE RPM platziert ist, und führen den folgenden Befehl in der Befehlszeile aus:

```
alien -d wingide-2.1.0-b2.i386.rpm
```

4) Eine `wingide-2.1.0-b2.deb` Datei ist jetzt im gleichen Verzeichnis zu finden. Um dieses Paket zu installieren führen Sie den folgenden Befehl in der Befehlszeile aus:

```
dpkg -i wingide-2.1.0-b2
```

Dies ist eine allgemeine Technik, die auch für andere Linux-RPMs funktioniert.

**Verwendung des systemweiten GTK:**

Wing IDE läuft standardmäßig mit seiner eigenen Kopie von GTK2 und nimmt das systemkonfigurierte Thema nicht auf. Dies wird gemacht, um Probleme und Fehler, die manchmal durch geringe Binärinkompatibilitäten in GTK-Versionen verursacht werden, zu vermeiden.

In Linux-Versionen, die GTK-Version 2.2 oder höher enthalten, können Sie Wing IDE auffordern, das systemdefinierte GTK2 zu verwenden, indem Sie die Einstellung **System-GTK** verwenden oder mit dem `--system-gtk` Argument der Command Line ausführen.

Die Verwendung des systemweiten GTK2 auf diese Weise funktioniert im Allgemeinen ziemlich gut, kann aber aufgrund von Binärinkompatibilitäten in GTK- und verwandten Bibliotheken zu Abstürzen oder Fehlern der Anzeige führen. Wenn Sie die Einstellung setzen und Wing nicht startet, müssen Sie in der Command Line die Option `--private-gtk` bestimmen, um die Einstellung zu überschreiben.

**Nicht-ASCII-Dateipfade auf älteren Linux-Systemen:**

Einige ältere Linux-Versionen erfordern das Setzen der Umgebungsvarible G_BROKEN_FILENAMES, bevor Wing IDE's Dialog Datei öffnen/speichern mit Dateipfaden, die nicht-ASCII-Zeichen enthalten, richtig funktioniert. Die Umgebungsvariable ist auf einigen Systemen, auf denen sie benötigt wird, bereits eingestellt; dies ist jedoch nicht immer der Fall.

### 1.12.3. Source-Code-Installation

Source-Code steht nur für lizenzierte Nutzer von Wing IDE Professional (nur Nicht-Bewertungslizenzen), die eine Geheimhaltungsvereinbarung unterzeichnet haben, zur Verfügung. Nach Erhalt dieser Vereinbarung stellen wir Ihnen Anweisungen zum Erlangen und Arbeiten mit dem Source-Code des Produktes bereit.

## 1.13. Wing IDE entfernen

#### Windows

In Windows verwenden Sie die Schaltfläche `Programme Hinzufügen/Entfernen`, wählen Wing IDE aus und entfernen es.

#### Linux/Unix

Zum Entfernen einer RPM-Installation in Linux müssen Sie `rpm -e wingide` eintippen.

Zum Entfernen einer tar-Archiv-Installation in Linux/Unix, müssen Sie das `wing-uninstall` Skript in `WINGHOME` aufrufen. Dieser Vorgang wird automatisch alle Dateien entfernen, die seit der Installation nicht geändert wurden. Sie werden danach gefragt, ob alle Dateien, die geändert wurden, entfernt werden sollen.

#### Mac OS X

Um Wing von Mac OS X zu entfernen, müssen Sie einfach seinen Anwendungsordner in den Papierkorb verschieben.

#### Eine Patch-Installation entfernen

Wenn Sie vorher Patch-Dateien auf Ihre Wing IDE Installation angewendet haben, müssen Sie verbleibende Dateien und Verzeichnisse nach der Deinstallation manuell entfernen. In diesem Fall sollten Sie Ihr **Verzeichnis der Benutzereinstellungen** aufbewahren, welches Lizenz- und Einstellungsinformationen enthält.

## 1.14. Verwendung der Befehlszeile

Immer wenn Sie `wing2.1` von der Command Line ausführen, sollten Sie eine Liste der zu öffnenden Dateien bestimmen. Dies können beliebige Textdateien oder eine Projektdatei sein. Das Folgende wird beispielsweise die Projektdatei `myproject.wpr` und auch die drei Quelldateien `mysource.py`, `README` und `Makefile` öffnen:

```
wing2.1 mysource.py README Makefile
```

Literal block ends without a blank line; unexpected unindent.

myproject.wpr

(in Windows wird die Executable `wing.exe` genannt)

Wing bestimmt den Dateityp nach dem Dateizusatz; daher ist die Position des Namens der Projektdatei (wenn vorhanden) in der Befehlszeile unwichtig.

Die folgenden zulässigen Optionen können überall in der Befehlszeile angegeben werden:

- **--prefs-file** -- Fügt den Dateinamen, der diesem Argument folgt, zu der Liste der Einstellungsdateien, die vom IDE geöffnet werden, hinzu. Diese Dateien werden nach den systemweiten Dateien und den Einstellungsdateien des Standard-Nutzers geöffnet, so dass ihre Werte die Werte von anderen Einstellungsdateien außer Kraft setzen.
- **--new** -- Wing wird standardmäßig eine bestehende, ausführende Instanz von Wing IDE wiederverwenden, um Dateien, die in der Befehlszeile bestimmt sind, zu öffnen. Diese Option schaltet dieses Verhalten aus und erzwingt die Erstellung einer neuen Instanz von Wing IDE. Beachten Sie, dass eine neue Instanz immer erstellt wird, wenn in der Befehlszeile keine Dateien angegeben sind.
- **--system-gtk** -- *(Nur Posix)* Diese Option führt dazu, dass Wing versucht, die systemweite Installation von GTK2 zu verwenden, anstatt seiner eigenen GTK-Version, ohne Berücksichtigung der Grundeinstellungen. Die Ausführung in diesem Modus verursacht, dass Wing systemweite Themenvoreinstellungen übernimmt. Aufgrund von Inkompatibilitäten in GTK- und verwandten Bibliotheken kann dies allerdings zu Abstürzen oder Problemen der Anzeige führen.
- **--private-gtk** -- *(Nur Posix)* Diese Option führt dazu, dass Wing seine private Kopie von GTK2 und verwandten Bibliotheken verwendet, ohne Berücksichtigung der Grundeinstellungen. Die Verwendung des privaten GTK kann dazu führen, dass Wing nicht das systemweite Thema übernimmt; es verhindert allerdings Inkompatibilitäten mit der systemweiten GTK-Bibliothek.
- **--verbose** -- *(Nur Posix)* Diese Option verursacht, dass Wing umfangreiche Fehlerberichtausgaben an `stderr` druckt. In Windows müssen Sie stattdessen `console_wing.exe` ausführen, um das gleiche Ergebnis zu erzielen.
- **--display** -- *(Nur Posix)* Stellt die X Windows-Anzeige ein, mit der Wing ausführen soll. Die Anzeigebestimmung sollte diesem Argument in Standard-Format folgen, z.B. `myhost:0.0`.

- **--use-winghome** -- *(Nur für Entwickler)* Diese Option stellt ein, dass WINGHOME während dieser Ausführung verwendet wird. Es wird intern und von Entwicklern, die zu Wing IDE beitragen, verwendet. Das zu verwendende Verzeichnis folgt diesem Argument.
- **--use-src** -- *(Nur für Entwickler)* Diese Option wird verwendet, um Wing zu zwingen, von Python Source-Dateien auszuführen, selbst wenn die kompilierten Dateien im `bin` Verzeichnis vorhanden sind, wie es der Fall ist, nachdem eine Distribution erstellt wurde.
- **--orig-python-path** -- *(Nur für Entwickler)* Diese Option wird intern verwendet, um den ursprünglichen Python-Pfad anzuzeigen, den der Benutzer genutzt hat, bevor Wing gestartet wurde. Der Pfad folgt diesem Argument.
- **--squelch-output** -- *(Nur für Entwickler)* Diese Option verhindert die Ausgabe jeglicher Art an `stdout` und `stderr`. In Windows wird sie verwendet, um die Konsolenerstellung zu verhindern.

## 1.15. Fehlerbehebung

Dieser Abschnitt beschreibt, was Sie tun können, wenn bei der Installation oder Anwendung von Wing IDE Probleme auftreten.

> Wir sind für Feedback und Fehlerberichte sehr dankbar. Beide können direkt von Wing IDE eingereicht werden, indem Sie die Einträge `Feedback einreichen` und `Fehlerbericht einreichen` aus dem Menü Hilfe verwenden oder uns eine E-Mail an support at wingware.com senden.

### 1.15.1. Fehlerbehebung für Startfehler

Wenn Sie Probleme haben, Wing zum Laufen zu bringen, dann lesen Sie diesen Abschnitt, um Informationen zur Diagnose des Problems zu erhalten.

1) In OS X erfordert Wing, dass Sie einen X11 Server installieren und starten, bevor Sie Wing IDE starten. Siehe das **OS X How-To** für Einzelheiten.
2) In OS X muss außerdem die .tar.gz Datei, in der Wing enthalten ist, mit dem StuffIt Expander oder mit dem gnutar Befehl (nicht mit dem tar Befehl) extrahiert werden. Dies muss aufgrund der Längenbeschränkung von Dateinamen in tar erfolgen.

3) In Windows wird das temporäre Verzeichnis des Nutzers manchmal voll, wodurch das Starten von Wing verhindert wird. Prüfen Sie, ob das Verzeichnis mehr als 65.534 Dateien enhält. Einige Versionen von Acrobat Reader lassen sehr viele Sperrdateien in diesem Verzeichnis. Diese Dateien heißen `Acr-xxxx.tmp`.

4) In Linux kann das Setzen der Einstellung **System-GTK verwenden** dazu führen, dass Wing in einigen Linux-System nicht startet. In diesem Fall müssen Sie die Datei `use-system-gtk` aus Ihrem **Verzeichnis der Benutzereinstellungen** entfernen, Wing starten und die Auswahl des Kontrollkästchens für die Einstellung `System-GTK verwenden` aufheben. Das Qt-Thema verursacht oft Abstürze und bei älteren Linux-Systemen können mit dieser Option Probleme auftreten.

5) Um Probleme mit einer Projektdatei oder Einstellungen auszuschließen, benennen Sie Ihr **Verzeichnis der Benutzereinstellungen** um und starten Wing neu. Wenn dies funktioniert, können Sie Dateien aus dem umbenannten Verzeichnis - eine nach der anderen - herüberkopieren, um das Problem zu isolieren. Sie können auch eine E-Mail an support at wingware dot com schicken, wenn Sie Hilfe benötigen.

6) Unter einem Windows Terminal-Server kann es sein, dass Wing nicht in der Lage ist, die Umgebungsvariablen, die es intern verwendet, zu setzen und wird daher nicht starten. In diesem Fall können Sie Wing mit den folgenden Befehlen zum Laufen bringen:

```
set PYTHONOPTIMIZE=1
set PYTHONHOME=D:\Program Files\WingIDE\bin\PyCore
wing.exe
```

   Ändern Sie `PYTHONHOME` entsprechend dem Speicherort, an dem Sie Wing IDE installiert haben.

7) In anderen Fällen lesen Sie bitte den Abschnitt **Diagnoseausgabe erhalten**.

## 1.15.2. Probleme in Microsoft Windows

Wing hat einige Probleme/Beschränkungen in Microsoft Windows Systemen.

1) Einige Demo-Shell Erweiterungs-COM-Objekte von win32all können Wing zum Abstürzen bringen, wenn sie registriert sind. Das Abstürzen passiert, wenn die Dialogboxen Datei öffnen, Speichern und Dateien zum Projekt hinzufügen verwendet werden. Diese Erweiterungen können mit ShellExView

(http://www.snapfiles.com/get/shellexview.html) deaktiviert werden; Sie können auch ein ähnliches Programm nutzen, um die Erweiterungen zu finden und zu deaktivieren. Die Erweiterungen können auch deinstalliert werden, indem die .py Datei mit einem `--unregister` Argument ausgeführt wird.

2) Der nVidia Desktop-Manager kann in einigen Windows-Versionen zu Abstürzen führen (die Grafikkarte scheint langsam zu werden, während die Ausnutzung des System-CPU ungefähr 0 % bleibt). Dieses Problem tritt am häufigsten auf, wenn Wing mit mehreren Fenstern verwendet wird, aber kann auch in allen anderen Fällen passieren. Die Deaktivierung des Managers verhindert das Abstürzen.

Es können auch andere Anzeigefehler auftreten (beispielsweise kann es passieren, dass der Fensterinhalt nicht angezeigt wird, wenn das Fenster von der Windows Menüleiste wiederhergestellt wird). Dies ist insbesondere für einige nVidida-Karten der Fall, selbst wenn der Desktop-Manager deaktiviert ist. Wir untersuchen das Problem und arbeiten an der Fehlerbehebung für zukünftige Versionen.

3) Windows Ziehen-und-Ablegen (Drag-n-Drop) funktioniert nicht für die Übertragung von Daten (Text oder Dateien) zwischen Wing und Windows Desktop oder anderen Anwendungen.

## 1.15.3. Fehlerbehebung für Debug-Fehler

Wenn Sie Probleme beim Debuggen mit Wing IDE haben, wählen Sie aus den folgenden Punkten denjenigen aus, der das von Ihnen beobachtete Problem am besten beschreibt:

- **Starten des Debug-Prozesses scheitert**
- **Debugger berichtet Exceptions, die außerhalb von Wing nicht gesehen werden**
- **Debugger stoppt nicht an Haltepunkten**
- **Debugger stoppt nicht an Exceptions**

### 1.15.3.1. Fehler beim Starten des Debug-Prozesses

In bestimmten Fällen kann Wing daran scheitern, den Debug-Prozess zu starten. Wenn dies passiert, ist es oft hilfreich, einen kleinen Test, wie den folgenden, zu debuggen:

```
print "test1"
print "test2"
```

Verwenden Sie den Befehl `Debuggen / Fortsetzen` aus dem Menü Debuggen, um Wing IDE zu dem Versuch zu veranlassen, nur bis zur ersten Zeile Ihres Codes auszuführen. Dies schließt mögliche Fehler aus, die durch spezifischen Code ausgelöst werden.

Prüfen Sie dann die folgenden allgemeinen Probleme. Für Informationen, wie Sie zusätzliche Informationen aus dem Debug-Untersystem erhalten, lesen Sie den Abschnitt **Diagnoseausgabe erhalten**:

1) Wing's Debugger verwendet ein TCP/IP-Protokoll, um mit dem IDE zu kommunizieren. Versichern Sie sich, dass TCP/IP auf Ihrem Computer installiert und konfiguriert ist. Dies ist manchmal ein Problem auf Windows 98, wenn zum Beispiel eine PCMCIA-Netzwerkkarte ausgeworfen wird.

2) Wenn Wing anzeigt, dass es Python nicht finden kann oder wenn Sie mehrere Versionen von Python auf Ihrem System haben, dann versichern Sie sich, dass Sie Ihre **Projekteigenschaften** so eingestellt haben, dass sie einen gültigen Interpreter enthalten (siehe Menüpunkt Source/Analysestatistiken anzeigen, um zu prüfen, dass der richtige Interpreter gefunden wird).

3) Geben Sie die notwendigen `PYTHONPATH` für Ihren Debug-Prozess in Projekteigenschaften ein, wenn nicht bereits in der Umgebung definiert.

4) Wenn Sie die Umgebungsvariablen `PYTHONHOME` oder `PYTHONPATH` einstellen, können diese das Scheitern des Debug-Prozesses verursachen, wenn sie nicht mit dem bestimmten Python-Interpreter, den Wing startet, übereinstimmen. Sie können entweder den verwendeten Interpreter wechseln, so dass es übereinstimmt, oder Sie ändern diese Umgebungswerte von außerhalb oder mit dem Punkt Projekteigenschaften aus dem Menü Projekt oder setzen diese Werte in der gleichen Weise zurück.

   `PYTHONHOME` ist ein Problem in allen Fällen, in denen es nicht mit dem Python-Interpreter, der im Menü Source unter dem Punkt `Analysestatistiken anzeigen` berichtet wird, übereinstimmt.

   `PYTHONPATH` ist nur ein Problem, wenn es Verzeichnisse enthält, die Teil einer Python-Installation sind. Wenn dies nicht mit der Version des Interpreters übereinstimmt, führt es zu Importfehlern, weil Python versucht, nicht kompatible Module zu importieren.

5) Überprüfen Sie in Windows, dass Sie den Hummingbird Socks Client nicht auf Ihrer Maschine installiert haben. Einige Versionen und Konfigurationen dieses Produkts sind dafür bekannt, dass sie Netzwerkpakete falsch weiterleiten, und zwar in einer Art und Weise, die den Wing IDE Debugger genug verlangsamt, um ihn während der Initialisierung abzuschalten.

6) Alle Formen der Python-Binärdistribution (TAR, RPM und Windows Installer) sind dafür bekannt, dass sie Probleme haben, wenn eine neuere Python-Version direkt über eine ältere Version auf dem Laufwerk installiert wird.

   In diesem Fall scheinen die meisten Python-Programme außerhalb von Wing IDE bestens zu funktionieren, aber werden innerhalb des Wing IDE Debuggers nicht funktionieren. Dies tritt auf, weil der Debug-Support-Code Sockets und andere Funktionalitäten verwendet, die von Ihrem Debug-Programm außerhalb des Wing Debuggers nicht notwendigerweise ausgeübt werden.

   Wenn Sie versuchen, eine Debug-Sitzung in Wing IDE auszuführen und es scheitert, werden Sie wahrscheinlich dieses Problem haben. Das folgende Test-Skript kann verwendet werden, um zu bestätigen, dass das Problem in Ihrer Python-Installation existiert (obwohl nicht bekannt ist, ob das Skript Exceptions in allen Fällen anzeigt, in denen eine fehlerhafte Python-Installation die Ursache des Debug-Problems ist):

   ```
   import sys
   print 'sys.version =', sys.version
   print 'sys.executable =', sys.executable
   print 'sys.version_info =', sys.version_info
   import socket
   print 'socket =', socket
   print 'socket._socket =', socket._socket
   import select
   print 'select =', select
   import cPickle
   print 'cPickle =', cPickle
   ```

   Um dieses Problem zu lösen, versuchen Sie, Python zu deinstallieren, entfernen alle verbleibenden Dateien manuell und installieren es dann wieder. Eine andere Möglichkeit ist, Python an einem neuen Speicherort auf der Festplatte zu installieren.

   Wenn dies einmal erledigt ist, vesichern Sie sich im Dialog Projekteingenschaften im Projektmenü, dass Wing konfiguriert ist, die neue Python-Installation zu verwenden, und das im Punkt `Analysestatistiken anzeigen` im Menü Source der richtige Interpreter angezeigt wird.

### 1.15.3.2. Zusätzliche Exceptions im Debugger

Wing's Debugger scheint manchmal Fehler aufzudecken, die nicht zu sehen sind, wenn außerhalb des Debuggers ausgeführt wird. Dies resultiert aus der Art und Weise, in der Wing entscheidet, welche Exceptions dem Nutzer angezeigt werden sollten. Wing

prüft Exceptions, wenn diese angetroffen werden, und entscheidet, ob die Exception unerwartet oder Teil der normalen Ausführung ist.

Sie können Wing trainieren, ungewünschte Berichte von Exceptions zu ignorieren, indem Sie das Kontrollkästchen im Werkzeug `Exceptions` anklicken.

Sie können auch die Art und Weise ändern, in der Wing Exceptions des Debug-Prozesses berichtet. Verwenden Sie dafür die Einstellung **Berichten von Exceptions**.

Für zusätzliche Informationen lesen Sie bitte den Abschnitt **Exceptions verwalten**.

#### 1.15.3.3. Fehler beim Stoppen an Exceptions

Standardmäßig stoppt Wing nur an Exceptions, von denen es denkt, dass sie unbehandelt sind. Wenn Ihr Code innerhalb einer Catch-all try/except-Klausel läuft, die in Python geschrieben ist (wie in einigen GUI-Hauptschleifen oder in einer Umgebung wie Zope), wird Wing keine Exceptions berichten, die in Ihrem Debug-Prozess angetroffen werden, außer wenn diese Exception dazu führt, dass der Debug-Prozess beendet wird.

Um Wing zum Anhalten zu bringen, können Sie die Einstellung **Berichten von Exceptions** auf `Immer sofort` setzen. Dies berichtet jedoch oft viele andere Exceptions, die intern während der normalen Ausführung auftreten.

Eine Alternative besteht darin, neuen Code für Ihre Anwendung zu schreiben, um den Catch-all Exception-Handler optional zu machen, wie in dem folgenden Beispiel:

```
import os

# Kein Handler bei der Ausführung in Wing's Debugger
if os.environ.has_key['WINGDB_ACTIVE']:
  dosomething()

# Unerwartete Exceptions zu anderer Zeit verarbeiten
else:
  try:
    dosomething()
  except:
    # Handler hier
```

Alternativ können Sie den folgenden Code zu Ihrem Catch-all Exception-Handler hinzufügen:

```
import os
```

```
if os.environ.has_key['WINGDB_ACTIVE']:
  raise
```

Dies wird Ihren Debug-Prozess an der Exception beenden, aber wird Ihnen nicht immer ermöglichen, den Programmzustand zu dem Zeitpunkt, an dem die Exception angetroffen wurde, zu überprüfen.

Beachten Sie, dass Umgebungen, wie wxPython, PyGTK und andere, Catch-all Handlers für unerwartete Exceptions, die in der Hauptschleife angetroffen werden, enthalten. Diese Handler sind allerdings in C/C++ Erweiterungsmodul-Code geschrieben und werden folglich von Wing ohne Änderungen am Handler korrekt berichtet.

#### 1.15.3.4. Debugger stoppt nicht an Haltepunkten oder zeigt Source-Code nicht an

Die häufigste Ursache für das Scheitern, an Haltepunkten zu stoppen oder Source-Fenster aufzuschlagen, während angehalten ist oder durch den Code geschritten wird, ist die Nichtübereinstimmung zwischen dem Dateinamen, der in der `*.pyc` Datei gespeichert ist, und der tatsächlichen Position der `*.py` Source-Datei.

Dies kann verursacht werden durch: (1) Nicht speichern bevor Sie im Debugger ausführen, (2) Verschieben der `*.pyc` Datei nachdem sie erstellt wurde oder (3) Verwendung von `compileall.py`, um `*.pyc` Dateien vom Source-Code zu erstellen.

Die einfachste Möglichkeit, Probleme mit `*.pyc` Dateien zu lösen, besteht darin, alle Ihre `*.pyc` Dateien vor dem Debuggen zu entfernen. Zum Beispiel mit dem folgenden Befehl in Linux/Unix:

```
rm -f ‘find . -name \*.pyc‘
```

**Wichtig:** Machen Sie dies nicht systemweit, sondern nur in den Verzeichnissen, die die entsprechenden `*.py` Source-Dateien für alle `*.pyc` Dateien enthalten. Einige Binärinstallationen von Python-Werkzeugen, einschließlich Teile von Wing IDE, enthalten nur die `*.pyc` Datei und keinen `*.py` Source-Code. In diesen Fällen führt das Entfernen von `*.pyc` zu `ImportError`, wenn Pyhton die `*.pyc` Datei nicht finden kann und sie aus dem Source-Code nicht neu erzeugen kann!

Ein weiteres, häufig auftretendes Problem ist das Ausführen des Debug-Prozesses mit einem teilweisen oder relativen Pfadnamen und die Verwendung von `os.chdir()` während der Ausführung. Dies kann den Debugger in bestimmten Fällen durcheinander bringen.

Es kann passieren, dass Wing nicht stoppt, wenn es eine Mehrpfadanwendung (Multi-Threaded Application) debuggt, weil der Debugger zur Zeit immer nur einen Pfad

(Thread) debuggen kann. Wenn in einem Pfad (Thread), außer dem Hauptpfad, ein Haltepunkt erreicht wird (oder bei einem extern gestarteten Prozess der Pfad (Thread), in den `wingdbstub` importiert wurde), wird der Haltepunkt ignoriert.

Weniger häufige Ursachen dieses Problems sind: (1) Die Ausführung von Python mit der `-O` Optimierungsoption, (2) das außer Kraft setzen der Python `__import__` Routine, (3) das Hinzufügen von Haltepunkten, nachdem Sie mit dem Debuggen einer Anwendung, die viel ihrer Zeit in C/C++ oder anderem nicht-Python-Code verbringt, begonnen haben, und (4) in win32, die Verwendung symbolischer Links zu Verzeichnissen, die Ihre Source-Code-Dateien enthalten (Posix-Plattformen verarbeiten symbolische Links problemlos).

Zusätzliche Informationen finden Sie im Abschnitt **Beschränkungen des Debuggers**.

## 1.15.4. Diagnoseausgabe erhalten

Wing IDE und Ihr Debug-Code laufen in separaten Prozessen, wobei beide unabhängig voneinander konfiguriert werden können, um zusätzliche diagnostische Protokollinformationen zu sammeln.

**Allgemeine IDE Probleme diagnostizieren**

Ein schneller Weg zum Diagnostizieren von Problemen, die beim Arbeiten mit Wing IDE gesehen werden, besteht darin, einen Fehlerbericht vom Menü `Hilfe` einzureichen. Bitte fügen Sie eine Beschreibung des Problems bei und klicken das Kontrollkästchen `Fehlerprotokoll aufnehmen` an, so dass wir das Problem ermitteln und beheben können.

Um andere Probleme, wie Fehler zu Starten, zu diagnostizieren, werfen Sie einen Blick auf die `error-log` Datei in Ihrem **Verzeichnis der Benutzereinstellungen**.

Alternativ können Sie den Befehl `console_wing.exe` (in Windows) oder `wing --verbose` (in Linux) von der Befehlszeile ausführen, um die diagnostische Ausgabe anzuzeigen.

Senden Sie diese Ausgabe zusammen mit Ihrer Systemversion, der Version von Wing IDE und anderen möglicherweise relevanten Details per E-Mail an support at wingware.com

**Debugger-Probleme diagnostizieren**

Um Debugger-Probleme zu diagnostizieren, setzen Sie die Einstellung **Protokolldatei der Debug-Internals** auf einen Wert außer `Keine Protokollierung` und schalten die Einstellungen **Externe Konsole verwenden** und **Externe Konsole wartet auf Beenden** an. Wenn Sie dies erneut versuchen, wird Wing eine Debug-Konsole mit Diagnosen anzeigen.

Alternativ können Sie auch folgendes tun: Kopieren Sie `wingdbstub.py` aus Ihrer Wing IDE Installation. Setzen Sie die Umgebungsvariable `WINGDB_LOGFILE` auf `<stderr>` oder auf den Namen einer Protokolldatei auf dem Laufwerk (oder ändern Sie `kLogFile` innerhalb von `wingdbstub.py`). Setzen Sie die Einstellung **Passives Hören aktivieren** auf Wahr. Versuchen Sie dann, das folgende Skript von Ihrer Befehlszeile zu starten:

```
import wingdbstub
print "test1"
print "test2"
```

Dieses Vorgehen druckt eine diagnostische Ausgabe, die in einigen Fällen wahrscheinlich einfacher zu erfassen ist.

Senden Sie diese Ausgabe per E-Mail an support at wingware.com. Fügen Sie außerdem die Inhalte der Datei `error-log` aus Ihrem **Verzeichnis der Benutzereinstellungen** bei. Weitere erforderliche Informationen sind Ihre Systemversion, die Version von Wing IDE und andere möglicherweise relevante Details.

### 1.15.5. Wing IDE beschleunigen

Wing sollte selbst auf relativ langsamer Hardware eine ansprechbare, gut aussehende Benutzeroberfläche präsentieren. In einigen Fällen kann Wing träge erscheinen:

1) Das erste Mal, wenn Sie eine Projektdatei einrichten, analysiert Wing alle Source-Dateien für den Source-Code-Browser und die Auto-Vervollständigungs-Einrichtungen. Während dieser Zeit werden die klassenorientierten Ansichten des Browsers nur die Source-Konstrukte von Dateien, von denen bereits Analyseinformationen erhalten wurden, anzeigen. Die Benutzeroberfläche kann auch träge erscheinen und Wing wird eine wesentliche Menge der CPU-Zeit verbrauchen.

   Um diesen Effekt in nachfolgenden Sitzungen zu begrenzen, speichert Wing seine Source-Analyse-Informationen auf der Festplatte in einem Cache innerhalb Ihres **Verzeichnisses der Benutzereinstellungen**.

   In großen Projekten kann jedoch selbst das Lesen dieses Cache und das Überprüfen von Dateien auf Aktualisierungen eine Weile dauern, wenn Wing das erste Mal gestartet wird. Der Prozess geschieht im Hintergrund nach dem Start und dauert 7-15 Sekunden pro 100.000 Code-Zeilen auf einem Celeron 400 Prozessor.

   In allen Fällen wird Wing diesen Prozess schließlich beenden und sollte zu dieser Zeit während normalem Bearbeiten und Debuggen fast kein CPU verbrauchen.

2) In wxPython und anderem Code, der `from xxx import *` Stil-Importe verwendet, kann der Auto-Vervöllständiger anfangs langsam erscheinen, da er viele hundert Symbole verarbeiten muss. Dies sollte jedoch nur das erste Mal, wenn er aufgerufen wird, passieren.

3) Einige Nutzer haben berichtet, dass der Hummingbird Socks Client für Windows ein wesentliches Verlangsamen des Debuggers verursacht, was scheinbar aus falschen Routine-TCP/IP-Paketen resultiert.

### 1.15.6. Fehlerbehebung öffnungs-Fehler der Dateinamen mit Leerzeichen

**In Windows:** Wenn Sie Windows Dateiarten oder `Öffnen Mit` verwenden, um zu veranlassen, dass Python-Dateien mit Wing geöffnet werden, setzen einige Versionen von Windows die falsche Befehlszeile für das Öffnen der Datei. Sie können dieses Problem beheben, indem Sie *regedt32.exe*, *regedit.exe* oder ein ähnliches Werkzeug verwenden, um die folgende Registrierungsposition zu bearbeiten:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\Applications\wing.exe\shell\open\command
```

Das Problem ist, dass der dort gespeicherten Assoziation Anführungszeichen um das *%1* Argument fehlen. Es sollte stattdessen folgendermaßen lauten:

```
"C:\Programme\Wing IDE\bin\wing.exe" "%1" %*
```

**In Linux:** KDE's Konqueror hat das gleiche Problem, dass Dateinamen, die von der Befehlszeile an Anwendungen, die an eine Dateiart gebunden sind, weitergegeben werden, nicht von Anführungszeichen umgeben sind, d.h. die Befehlszeile wird nicht korrekt analysiert. Zur Zeit haben wir keine Lösung für dieses Problem.

# Anpassung

Es gibt viele Mögichkeiten, Wing IDE auf Ihre Bedürfnisse und Wünsche anzupassen. Dieses Kapitel beschreibt die Optionen, die für die persönliche Anpassung Ihrer Wing IDE Installation zur Verfügung stehen.

**Die folgenden Anpassungsmöglichkeiten stehen zur Verfügung:**

- Der Inhalt, das Layout und das Aussehen der IDE-Fenster können konfiguriert werden.
- Viele andere Optionen sind in den Einstellungen verfügbar.
- Der Editor kann mit verschiedenen Individualitäten ausgeführt werden (Emacs oder Standard).
- Tastaturkürzel können für jeden beliebigen Wing-Befehl hinzugefügt, entfernt oder geändert werden.
- Datei-Sets können definiert werden, um einige der IDE-Funktionen zu steuern.

## 2.1. Optionen der Benutzeroberfläche

Wing stellt eine Vielzahl von Optionen bereit, mit denen Sie die Benutzeroberfläche auf Ihre Bedürfnisse anpassen können. Einstellungen können gewählt werden, um die Anzahl und die Art der Fenster, die das IDE verwendet, auszuwählen, das Layout von Werkzeugen innerhalb der Fenster zu bestimmen, die Schriftart und -größe des angezeigten Textes, die Art und den Inhalt der Werkzeugleiste sowie das insgesamte Aussehen oder „Thema“ festzulegen.

### 2.1.1. Fensteraufteilungen

Wing IDE kann in einer Vielzahl von Fenstermodi ausgeführt werden. Dies wird mit der Einstellung **Fensteraufteilung** gesteuert, welche die folgenden Optionen bereitstellt:

- **Kombinierte Werkzeugbox und Editorfenster** -- Dies ist die Voreinstellung, bei welcher Wing ein einziges Fenster öffnet, in welchem der Editorbereich mit zwei Werkzeugboxfeldern kombiniert wird.
- **Separate Werkzeugboxfenster** -- In diesem Modus verschiebt Wing IDE alle Werkzeuge in ein separates Fenster, das von allen Werkzeugen gemeinsam genutzt wird.
- **Ein Fenster pro Editor** -- In diesem Modus erstellt Wind IDE ein Top-Level Fenster für jeden geöffneten Editor. Zusätzlich werden alle Werkzeuge in ein separates, gemeinsam genutztes Werkzeugboxfenster verschoben und die Werkzeugleiste und Menüs werden in ein gemeinsam genutztes Werkzeugleisten-/Menüfenster verschoben.

Die Fensteraufteilung wird verwendet, um die Anfangskonfiguration und grundsätzliche Aktionen von Fenstern im IDE zu beschreiben. Wenn sie geändert wird, konfiguriert Wing IDE Ihre Projekte neu, damit sie das erste Mal, wenn sie mit der neuen Einstellung verwendet werden, mit der Fensteraufteilung übereinstimmen.

Es ist allerdings auch möglich, zusätzliche IDE Fenster zu erstellen und Editoren und Werkzeuge in ein anderes Fenster und zwischen bestehenden Fenstern zu verschieben, ohne dabei die voreingestellte Fensteraufteilung zu verändern. Dies wird weiter unten beschrieben.

### 2.1.2. Layout der Benutzeroberfläche

Wenn Sie mit der voreingestellten Fensteraufteilung arbeiten, dann besteht der Hauptbereich der Benutzeroberfläche aus zwei Werkzeugboxen (standardmäßig im unteren Teil und auf der rechten Seite, aber dies kann in den **Einstellungen** geändert werden) und einem Bereich für Source-Editoren und die integrierte Hilfe.

Das Klicken auf einen bereits aktiven Notizbuchreiter führt dazu, dass Wing das gesamte Feld minimiert, so dass nur noch die Notizbuchreiter sichtbar sind. Ein erneuter Klick bringt die Werkzeugbox wieder zu ihrer ursprünglichen Größe zurück. Die Tasten F1 und F2 wechseln zwischen diesen Modi.

In anderen Fenstermodi werden die Werkzeugboxen und der Editorbereich in separaten Fenstern präsentiert, aber sie haben viele der unten beschriebenen Konfigurationsoptionen gemeinsam.

### Konfiguration der Werkzeugleiste

Die Konfigurationsoptionen für Wing's Werkzeugleiste ermöglichen Ihnen, die Größe und die Art der Werkzeugleistensymbole zu ändern. Außerdem können Sie bestimmen, ob zusätzlich oder anstelle der Symbole Text angezeigt werden soll. Sie steuern diese Optionen mit den Einstellungen **Größe der Werkzeugleistensymbole** und **Art der Werkzeugleistensymbole**.

Die Werkzeugleiste kann auch vollständig versteckt werden. Verwenden Sie dafür die Einstellung **Werkzeugleiste anzeigen**.

### Konfiguration des Editorbereiches

Das Popup-Menü Optionen in der oberen rechten Ecke des Editorbereiches ermöglicht Ihnen, den Editor in mehrere unabhängige Felder zu teilen oder mehrere Felder zusammenzufügen. Diese können waagerecht, senkrecht oder in einer beliebigen Kombination von diesen angeordnet werden. Bei mehreren vorliegenden Feldern sind alle innerhalb des Fenters geöffneten Dateien in jedem einzelenen Feld verfügbar. Dies ermöglicht Ihnen, an jeder beliebigen Auswahl von Dateien und/oder in verschiedenen Teilen derselben Datei zu arbeiten.

Das Popup-Menü Optionen kann auch verwendet werden, um zwischen Editoren mit Reitern und Editoren, die ein Popup-Menü für die Auswahl der Dateien anzeigen, zu wechseln (letzteres kann bei einer großen Anzahl von Dateien leichter zu handhaben sein). Außerdem können Sie Editoren in separate Fenster oder zwischen bestehenden Fenstern, wenn mehrere Fenster geöffnet sind, verschieben.

### Konfiguratin der Werkzeugboxen

Auch jede Werkzeugbox kann entlang der Axen des Notizbuches in jede beliebige Anzahl von Unterfeldern geteilt oder zusammengefügt werden. Klicken Sie dafür auf das Drop-Down-Symbol Optionen im Reiterbereich der Notizbücher (ein rechter Mausklick funktioniert auch). Die Anzahl der Werkzeugboxteilungen, die Wing standardmäßig anzeigt, hängt von der Größe Ihres Bildschrims ab.

Das Popdown-Menü Optionen kann auch zur Duplizierung von Werkzeugen oder zum Verschieben von Werkzeugen zwischen den Teilbereichen oder in separate Fenster verwendet werden.

Alle verfügbaren Werkzeuge werden im Menü Werkzeuge aufgezählt. Dieses Menü zeigt das zuletzt verwendete Werkzeug dieser Art an oder es fügt ein Werkzeug zu Ihrem Fenster an seinem voreingestellten Ort hinzu, wenn es noch nicht vorhanden ist.

### Zusätzliche Fenster erstellen

Zusätzlich zum Verschieben von bestehenden Editoren oder Werkzeugen in neue Fenster ist es auch möglich, neue Werkzeugfenster (anfänglich mit einem Werkzeug) und neue

Dokumentfenster (mit Editor und Werkzeugleiste, wenn es auf die gewählte Fensteraufteilung zutrifft) zu erstellen. Diese Optionen finden Sie im Menü Fenster.

Wing IDE wird den Zustand von all Ihren Fenstern als Teil der Projektdatei speichern, so dass das gleiche Fensterlayout und die gleichen Fensterinhalte in nachfolgenden Sitzungen wiederhergestellt werden.

### 2.1.3. Änderung der Textanzeige

Wing versucht, für jedes System, auf dem es ausgeführt wird, eine entsprechende Schriftart für die Anzeige zu finden. Viele Nutzer möchten jedoch sicherlich die Schriftart und -größe, die im Editor und anderen Bereichen der Benutzeroberfläche verwendet werden, auf die persönlichen Bedürfnisse anpassen. Sie können dies mit den Einstellungen **Schriftart/-größe des Source-Codes** und **Schriftart/-größe der Anzeige** vornehmen.

Die Konfiguration von Farbe und Schriftart der Syntax-Markierung ist derzeit nicht möglich, ohne dabei den Source-Code des IDE's zu verändern. Wing bietet allerdings die Möglichkeit, die Hintergrundfarbe des Editors einzustellen (mit der Einstellung **Hintergrund des Source-Codes**) und wird entsprechend dem ausgewählten Hintergrund angemessen sichtbare Farben für die Syntax-Markierung bestimmen.

Die Farbe, die für die Textmarkierung verwendet wird, kann auch festgelegt werden. Verwenden Sie dafür die Einstellung **Farbe der Textmarkierung**.

Änderungen der Farbeinstellung hängen oft vom ausgewählten, gesamtheitlichen Anzeigethema ab. Dies wird im nächsten Abschnitt erläutert.

### 2.1.4. Einstellung des insgesamten Anzeigethemas

Wing IDE basiert auf GTK2, einem auf mehreren Betriebssystemen funktionierendes Benutzeroberflächen-Toolkit, das anpassbare `Themen` bereitstellt, die das insgesamte Look & Feel der Benutzeroberfläche bestimmen. Wing's Standardthema variiert je nach Plattform (in Windows wird ein Windows Emulationsthema verwendet und in OS X wird ein OS X ähnliches Thema genutzt). Das Thema kann mit der Einstellung **Anzeigethema** geändert werden.

In den meisten Fällen wird das neue Thema sofort auf Wing's Benutzeroberfläche angewendet. Wenn Sie zur Standardeinstellung zurückwechseln möchten, kann in einigen Fällen ein Neustart erforderlich sein, was mit einer Nachrichtendialogbox angezeigt wird.

Einige Systeme mit langsameren Grafikkarten können mit den farbreicheren 3D-Themen

nicht so gut funktionieren. In diesem Fall ist die Verwendung von `Gtk-Standard` die beste Option, da es keine zusätzliche Grafikverarbeitung umfasst.

**System-GTK auf Linux**

Auf Linux-Systemen mit GTK 2.2 oder höher ist es möglich, dass Wing mit der systemweiten GTK-Installation und systemdefinierten Themen ausgeführt wird. Dies wird mit der Einstellung **System-GTK verwenden** oder mit den **Argumenten der Command Line** `--system-gtk` sowie `--private-gtk` gesteuert. Wing funktioniert mit den meisten 2.4.x GTK2-Releases ziemlich gut, aber es können trotzdem noch Probleme auftreten. Sollten Sie Probleme mit der Stabilität von Wing haben oder Funktionsstörungen der Anzeige beobachten, empfehlen wir Ihnen, die private GTK-Option zu verwenden.

## 2.2. Einstellungen

Wing besitzt viele Einstellungen, die die Funktionen des Editors, Debuggers, Source-Browsers, Projektmanagers und anderer Werkzeuge steuern.

Verwenden Sie den Eintrag `Einstellungen` im Menü `Bearbeiten`, um diese Einstellungen zu ändern. Dies ordnet alle verfügbaren Einstellungen nach Kategorie und stellt Zugriff auf die Dokumentation in Werkzeug-Tipps bereit, die angezeigt wird, wenn Sie mit der Maustaste über den beschrifteten Bereich links neben jeder Einstellung fahren. Alle Nicht-Standardwerte, die Sie mit dem **Einstellungsdialog** auswählen, werden in der Einstellungsdatei in Ihrem **Verzeichnis der Benutzereinstellungen** gespeichert.

### 2.2.1. Ebenen der Einstellungsdatei

Wing's Einstellungsmanager läuft auf einem geschichteten Set von Einstellungsdateien. Eine installationsweite Einstellungsdatei kann innerhalb von `WINGHOME` platziert werden und einzelne Nutzer können diese Werte außer Kraft setzen, indem sie eine Einstellungsdatei in Ihrem **Verzeichnis der Benutzereinstellungen** platzieren. Die Werte der nutzerspezifischen Einstellungsdatei haben Vorrang gegenüber allen Werten in der Standard `WINGHOME/preferences` Datei.

Es ist außerdem möglich, zusätzliche Einstellungsdateien in der Befehlszeile mit der `--prefs-file` Option zu bestimmen. Zum Beispiel:

```
wing2.1 --prefs-file /path/to/myprefs
```

Jede Datei, die auf diese Weise bestimmt wird, setzt die Werte, die in den pro-Nutzer oder installationsweiten Einstellungsdateien gespeichert sind, außer Kraft.

### 2.2.2. Format der Einstellungsdatei

Obwohl wir empfehlen, den grafischen Einstellungsmanager für die Änderung von Einstellungen zu verwenden, werden einige Nutzer wünschen, die zugrunde liegenden Textdateien direkt zu bearbeiten.

Das Format der Einstellungsdatei besteht aus einer Reihe von Abschnitten, die durch eingeklammerte Kopfzeilen getrennt sind. Zur Zeit ist `[user-preferences]` der einzig gültige Abschnitt und alle anderen Abschnitte werden ignoriert. Der Körper jedes Abschnittes ist eine Folge von Zeilen, von denen jede ein `name=value` Paar ist.

Jeder Einstellungsname ist in der Form *domain.preference*, wobei *domain* das beeinflusste IDE-Teilsystem und *preference* der Name der spezifischen Einstellung ist (zum Beispiel: `edit.personality` definiert die Laufzeit-Individualität des Source-Editors).

Als Einstellungswerte kann jeder beliebige Python-Ausdruck ausgewählt werden, der zu einer Zahl, einer Zeichenkette, einem Tuple, List oder Dictionary bewertet. Zusätzlich werden die Konstanten `true` und `false` definiert und als gültige Werte unterstützt. Lange Zeilen können durch das Platzieren eines Backslashs am Ende einer Zeile fortgesetzt werden und Kommentare können überall auf einer Zeile gesetzt werden, indem ihnen ein `#` vorangestellt wird.

Wenn Sie Einstellungsdateien per Hand schreiben möchten, werfen Sie einen Blick auf die **Einstellungsreferenz** für eine Dokumentation zu allen verfügbaren Einstellungen.

## 2.3. Editor-Individualitäten

Die voreingestellte Editor-Individualität für Wing implementiert die allgemein üblichen Tastaturkombinationen, die in einem einfachen, grafischen Texteditor zu finden sind. Diese nutzt für die Interaktion mit dem Editor hauptsächlich die grafische Benutzeroberfläche und begrenzt bei der Interaktion die Verwendung von komplexen, tastaturgesteuerten Befehlen.

**Emacs-Individualität**
Die erste Sache, die ein Emacs-Nutzer anstrebt, ist es, die Editor-Individualität so einzustellen, dass sie Emacs emuliert. Dies wird mit der Einstellung **Tastatur / Individualität** gemacht.

Mit der Emacs-Individualität können Tastenkombinationen verwendet werden, um den größten Teil der Editor-Funktionalität zu steuern. Es wird eine Dialogzeile für die Textinteraktionen ('Mini-Buffer') am unteren Ende des Editor-Fensters verwendet, in der

normalerweise die aktuelle Zeilennummer und andere informative Nachrichten angezeigt werden.

Es ist auch möglich, innerhalb jeder dieser Individualitäten individuelle Tastaturkürzel hinzuzufügen, zu ändern oder zu entfernen. Siehe **Tastaturkombinationen** für Einzelheiten.

## 2.4. Tastaturbefehle

Für Befehle, die mit der Tastatur aufgerufen werden, kann die Tastaturkombination geändert werden. Es ist dafür eine andere Datei mit Tastaturbefehlen zu bestimmen oder es können benutzerdefinierte Tastaturbefehle festgelegt werden. Ein benutzerdefinierter Tastaturbefehl überschreibt die in den Dateien für Tastaturbefehle festgelegten Einstellungen. Benutzerdefinierte Tastaturbefehle können mit der Einstellung **Benutzerdefinierte Tastaturbefehle** bestimmt werden.

Zum Hinzufügen eines Tastaturbefehls auf die Schaltfläche Einfügen klicken und dann im Feld `Taste` die Tastenkombination und im Feld `Befehl` den auszuführenden Befehl eingeben.

Tasturbefehle, die standardmäßig definiert sind oder von dieser Einstellung überschrieben werden, werden in allen Menüeinträgen, die den gleichen Befehl implementieren, angezeigt. Wenn einem Befehl mehrere Tastaturkombinationen zugewiesen wurden, wird nur die zuletzt gefundenen Kombination angezeigt. (Es funktionieren allerdings alle zugewiesenen Kombinationen mit der Tastatur.)

**Dateien für Tastaturbefehle**

Wing wird mit zwei Dateien für Tastaturbefehle geliefert, die beide in `WINGHOME` zu finden sind: `keymap.normal` und `keymap.emacs`. Diese werden als die Standard-Tastaturbefehle für die entsprechenden Editor-Individualitäten verwendet.

Es besteht die Möglichkeit, eine benutzerdefinierte Datei mit Tastaturbefehlen zu erstellen und diese mit der Einstellung **Datei für Tastaturbefehle** als Standarddatei festzulegen.

In einer Datei für Tastaturbefehle wird jede Tastaturkombination aus Namen, die im Abschnitt **Tastennamen** aufgelistet sind, gebildet. Diese Namen können folgendermaßen kombiniert werden:

1) Eine einzelne, unveränderte Taste wird allein durch ihren Namen bestimmt, zum Beispiel `'Down'` für die Pfeiltaste nach unten.
2) Bei veränderten Tasten werden die Tastennamen mit einem Bin-

destrich gekoppelt, zum Beispiel `'shift-Down'` für das Betätigen der Pfeiltaste nach unten während die Umschalttaste gedrückt ist. Mehrfache Modifikatoren können auch bestimmt werden, wie `'ctrl-shift-Down'`.

3) Mehrfach-Tastenkombinationen können bestimmt werden, indem mehrere Tastennamen mit einem Leerzeichen getrennt aufgelistet werden. Um beispielsweise eine Tastenkombination zu definieren, die daraus besteht, dass zuerst **ctrl-x** gedrückt wird und dann die **a** Taste allein gedrückt wird, verwenden Sie `'ctrl-x a'` als Tastenfolge.

Der Befehlsteil der Definition der Tastaturkombinationen kann einer der Befehle sein, die im Abschnitt **Befehlsreferenz** aufgelistet sind. Verwenden Sie None, um die gegebene Tastenkombination vollständig zu entfernen.

Wenn Sie eine Tastaturkombination festlegen, die bereits in den voreingestellten Tastaturbefehlen existiert, wird diese Kombination einfach mit Ihrem Wert ersetzt.

- **Beispiele**

Hier ist ein Beispiel für das Hinzufügen einer Tastenkombination für einen Befehl. Wenn der Befehl bereits einen voreingestellten Tastenbefehl hat, dann werden beide Kombinationen funktionieren:

```
'Ctrl-X P': 'debug-attach'
```

Dieses Beispiel entfernt eine Tastenkombination vollständig:

```
'Ctrl-C Ctrl-C': None
```

Diese können miteinander kombiniert werden, um die Tastenkombination für einen Befehl zu ändern, ohne die voreingestellte Tastenkombination zu behalten:

```
'Ctrl-C Ctrl-C': None
'Ctrl-G': 'debug-continue'
```

Wing behält immer nur den letzten Tastaturbefehl für eine gegebene Tastenkombination. Dieses Beispiel bindet Ctrl-X an 'quit' und keinen anderen Befehl:

```
'Ctrl-X': 'debug-stop'
'Ctrl-X': 'quit'
```

### 2.4.1. Tastennamen

- Tastenmodifikatoren, die von Wing IDE für Tastenkombinationen unterstützt werden, sind:

**Ctrl** -- Beide Steuerungstasten.

**Shift** -- Beide Umschalttasten. Dieser Modifikator wird mit einigen Tastennamen ignoriert, wie unten beschrieben.

**Alt** -- Nicht für den allgemeinen Gebrauch empfohlen, da diese Kombinationen oft mit Menü-Beschleunigern und Betriebssystem- oder Fenstermanager-Operationen in Konflikt zu stehen.

- Die numerischen Tasten und die Haupttasten des westlichen Alphabetes sind folgendermaßen bestimmt:

  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z,

- Diese Sondertasten können auch verwendet werden:

  Escape, Space, BackSpace, Tab, Linefeed, Clear, Return, Pause, Scroll_Lock, Sys_Req, Delete, Home, Left, Up, Right, Down, Prior, Page_Up, Next, Page_Down, End, Begin, Select, Print, Execute, Insert, Undo, Redo, Menu, Find, Cancel, Help, Break, Mode_switch, script_switch, Num_Lock,

  F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, L1, F12, L2, F13, L3, F14, L4, F15, L5, F16, L6, F17, L7, F18, L8, F19, L9, F20, L10, F21, R1, F22, R2, F23, R3, F24, R4, F25, R5, F26, R6, F27, R7, F28, R8, F29, R9, F30, R10, F31, R11, F32, R12, F33, R13, F34, R14, F35, R15,

- Für Äquivalente, die mit den Maustasten funktionieren, verwenden Sie diese:

  Pointer_Left, Pointer_Right, Pointer_Up, Pointer_Down, Pointer_UpLeft, Pointer_UpRight, Pointer_DownLeft, Pointer_DownRight, Pointer_Button_Dflt, Pointer_Button1, Pointer_Button2, Pointer_Button3, Pointer_Button4, Pointer_Button5, Pointer_DblClick_Dflt, Pointer_DblClick1, Pointer_DblClick2, Pointer_DblClick3, Pointer_DblClick4, Pointer_DblClick5, Pointer_Drag_Dflt, Pointer_Drag1, Pointer_Drag2, Pointer_Drag3, Pointer_Drag4, Pointer_EnableKeys, Pointer_Accelerate, Pointer_DfltBtnNext, Pointer_DfltBtnPrev,

- Die Tasten der Kleintastatur sind so bestimmt:

  KP_Left, KP_Right, KP_Up, KP_Down, KP_Home, KP_Page_Up, KP_Page_Down, KP_End, KP_Insert, KP_Delete, KP_0, KP_1, KP_2, KP_3, KP_4, KP_5, KP_6, KP_7, KP_8, KP_9,

- Diese funktionieren auch, aber ignorieren den Shift-Modifikator, da sie gewöhnlich auf internationalen Tastaturen an verschiedenen Stellen erscheinen:

  KP_Space, KP_Tab, KP_Enter, KP_F1, KP_F2, KP_F3, KP_F4, KP_Prior, KP_Next, KP_Begin, KP_Insert, KP_Delete, KP_Equal, KP_Multiply, KP_Add, KP_Separator, KP_Subtract, KP_Decimal, KP_Divide,

  exclam, quotedbl, numbersign, dollar, percent, ampersand, apostrophe, quoteright, parenleft, parenright, asterisk, plus, comma, minus, period, slash, colon, semicolon, less, equal, greater, question, at, bracketleft, backslash, bracketright, asciicircum, underscore, grave, quoteleft, braceleft, bar, braceright,

  EuroSign, EcuSign, ColonSign, CruzeiroSign, FFrancSign, LiraSign, MillSign, NairaSign, PesetaSign, RupeeSign, WonSign, NewSheqelSign, DongSign,

- Viele andere Tastennamen sind für internationale Tastaturen oder für Tastaturen für spezielle Zwecke verfügbar:

  asciitilde, nobreakspace, exclamdown, cent, sterling, currency, yen, brokenbar, section, diaeresis, copyright, ordfeminine, guillemotleft, notsign, hyphen, registered, macron, degree, plusminus, twosuperior, threesuperior, acute, mu, paragraph, periodcentered, cedilla, onesuperior, masculine, guillemotright, onequarter, onehalf, threequarters, questiondown,

  leftradical, topleftradical, horizconnector, topintegral, botintegral, vertconnector, topleftsqbracket, botleftsqbracket, toprightsqbracket, botrightsqbracket, topleftparens, botleftparens, toprightparens, botrightparens, leftmiddlecurlybrace, rightmiddlecurlybrace, topleftsummation, botleftsummation, topvertsummationconnector, botvertsummationconnector, toprightsummation, botrightsummation, rightmiddlesummation, lessthanequal, notequal, greaterthanequal, integral, therefore, variation, infinity, nabla, approximate, similarequal, ifonlyif, implies, identical, radical, includedin, includes, intersection, union, logicaland, logicalor, partialderivative, function, leftarrow, uparrow, rightarrow, downarrow, blank, soliddiamond, checkerboard, ht, ff, cr, lf, nl, vt, lowrightcorner, uprightcorner, upleftcorner, lowleftcorner, crossinglines, horizlinescan1, horizlinescan3, horizlinescan5, horizlinescan7, horizlinescan9, leftt, rightt, bott, topt, vertbar, emspace, enspace, em3space, em4space, digitspace, punctspace, thinspace, hairspace, emdash, endash, signifblank, ellipsis, doubbaselinedot, onethird, twothirds, onefifth, twofifths, threefifths, fourfifths, onesixth, fivesixths, careof, figdash, leftanglebracket, decimalpoint, rightanglebracket, marker, oneeighth, threeeighths, fiveeighths, seveneighths, trademark, signaturemark, trademarkincircle, leftopentriangle, rightopentriangle, emopencircle, emopenrectangle, leftsinglequotemark, rightsinglequotemark, leftdoublequotemark, rightdoublequotemark, prescription, minutes, seconds, latincross, hexagram, filledrectbullet, filledlefttribullet, filledrighttribullet, emfilledcircle, emfilledrect, enopencircbullet, enopensquarebullet, openrectbullet, opentribulletup, opentribulletdown, openstar, enfilledcircbullet, enfilledsqbullet, filledtribulletup, filledtribulletdown, leftpointer, rightpointer, club, diamond, he-

art, maltesecross, dagger, doubledagger, checkmark, ballotcross, musicalsharp, musicalflat, malesymbol, femalesymbol, telephone, telephonerecorder, phonographcopyright, caret, singlelowquotemark, doublelowquotemark, cursor, leftcaret, rightcaret, downcaret, upcaret, overbar, downtack, upshoe, downstile, underbar, jot, quad, uptack, circle, upstile, downshoe, rightshoe, leftshoe, lefttack, righttack,

Multi_key, Codeinput, SingleCandidate, MultipleCandidate, PreviousCandidate, Kanji, Muhenkan, Henkan_Mode, Henkan, Romaji, Hiragana, Katakana, Hiragana_Katakana, Zenkaku, Hankaku, Zenkaku_Hankaku, Touroku, Massyo, Kana_Lock, Kana_Shift, Eisu_Shift, Eisu_toggle, Kanji_Bangou, Zen_Koho, Mae_Koho,

ISO_Lock, ISO_Level2_Latch, ISO_Level3_Shift, ISO_Level3_Latch, ISO_Level3_Lock, ISO_Group_Shift, ISO_Group_Latch, ISO_Group_Lock, ISO_Next_Group, ISO_Next_Group_Lock, ISO_Prev_Group, ISO_Prev_Group_Lock, ISO_First_Group, ISO_First_Group_Lock, ISO_Last_Group, ISO_Last_Group_Lock, ISO_Left_Tab, ISO_Move_Line_Up, ISO_Move_Line_Down, ISO_Partial_Line_Up, ISO_Partial_Line_Down, ISO_Partial_Space_Left, ISO_Partial_Space_Right, ISO_Set_Margin_Left, ISO_Set_Margin_Right, ISO_Release_Margin_Left, ISO_Release_Margin_Right, ISO_Release_Both_Margins, ISO_Fast_Cursor_Left, ISO_Fast_Cursor_Right, ISO_Fast_Cursor_Up, ISO_Fast_Cursor_Down, ISO_Continuous_Underline, ISO_Discontinuous_Underline, ISO_Emphasize, ISO_Center_Object, ISO_Enter

dead_grave, dead_acute, dead_circumflex, dead_tilde, dead_macron, dead_breve, dead_abovedot, dead_diaeresis, dead_abovering, dead_doubleacute, dead_caron, dead_cedilla, dead_ogonek, dead_iota, dead_voiced_sound, dead_semivoiced_sound, dead_belowdot,

First_Virtual_Screen, Prev_Virtual_Screen, Next_Virtual_Screen, Last_Virtual_Screen, Terminate_Server, AccessX_Enable, AccessX_Feedback_Enable, RepeatKeys_Enable, SlowKeys_Enable, BounceKeys_Enable, StickyKeys_Enable, MouseKeys_Enable, MouseKeys_Accel_Enable, Overlay1_Enable, Overlay2_Enable, AudibleBell_Enable, Pointer_Left, Pointer_Right, Pointer_Up,

_3270_Duplicate, _3270_FieldMark, _3270_Right2, 3270_Left2, _3270_BackTab, __3270_EraseEOF, _3270_EraseInput, 3270_Reset, _3270_Quit, _3270_PA1, __3270_PA2, _3270_PA3, _3270_Test, 3270_Attn, _3270_CursorBlink, __3270_AltCursor, _3270_KeyClick, 3270_Jump, _3270_Ident, _3270_Rule, __3270_Copy, _3270_Play, _3270_Setup, 3270_Record, _3270_ChangeScreen, __3270_DeleteWord, _3270_ExSelect, 3270_CursorSelect, _3270_PrintScreen, __3270_Enter,

Agrave, Aacute, Acircumflex, Atilde, Adiaeresis, Aring, AE, Ccedilla, Egrave, Eacute, Ecircumflex, Ediaeresis, Igrave, Iacute, Icircumflex, Idiaeresis, ETH, Eth, Ntilde, Ograve, Oacute, Ocircumflex, Otilde, Odiaeresis, multiply, Ooblique, Ugrave, Uacute, Ucircumflex, Udiaeresis, Yacute, THORN, Thorn, ssharp, agrave, aacute, acircumflex, atilde, adiaeresis, aring, ae, ccedilla, egrave, eacute, ecircumflex,

ediaeresis, igrave, iacute, icircumflex, idiaeresis, eth, ntilde, ograve, oacute, ocircumflex, otilde, odiaeresis, division, oslash, ugrave, uacute, ucircumflex, udiaeresis, yacute, thorn, ydiaeresis, Aogonek, breve, Lstroke, Lcaron, Sacute, Scaron, Scedilla, Tcaron, Zacute, Zcaron, Zabovedot, aogonek, ogonek, lstroke, lcaron, sacute, caron, scaron, scedilla, tcaron, zacute, doubleacute, zcaron, zabovedot, Racute, Abreve, Lacute, Cacute, Ccaron, Eogonek, Ecaron, Dcaron, Dstroke, Nacute, Ncaron, Odoubleacute, Rcaron, Uring, Udoubleacute, Tcedilla, racute, abreve, lacute, cacute, ccaron, eogonek, ecaron, dcaron, dstroke, nacute, ncaron, odoubleacute, udoubleacute, rcaron, uring, tcedilla, abovedot, Hstroke, Hcircumflex, Iabovedot, Gbreve, Jcircumflex, hstroke, hcircumflex, idotless, gbreve, jcircumflex, Cabovedot, Ccircumflex, Gabovedot, Gcircumflex, Ubreve, Scircumflex, cabovedot, ccircumflex, gabovedot, gcircumflex, ubreve, scircumflex, kra, kappa, Rcedilla, Itilde, Lcedilla, Emacron, Gcedilla, Tslash, rcedilla, itilde, lcedilla, emacron, gcedilla, tslash, ENG, eng, Amacron, Iogonek, Eabovedot, Imacron, Ncedilla, Omacron, Kcedilla, Uogonek, Utilde, Umacron, amacron, iogonek, eabovedot, imacron, ncedilla, omacron, kcedilla, uogonek, utilde, umacron, OE, oe, Ydiaeresis, overline,

kana_fullstop, kana_openingbracket, kana_closingbracket, kana_comma, kana_conjunctive, kana_middledot, kana_WO, kana_a, kana_i, kana_u, kana_e, kana_o, kana_ya, kana_yu, kana_yo, kana_tsu, kana_tu, prolongedsound, kana_A, kana_I, kana_U, kana_E, kana_O, kana_KA, kana_KI, kana_KU, kana_KE, kana_KO, kana_SA, kana_SHI, kana_SU, kana_SE, kana_SO, kana_TA, kana_CHI, kana_TI, kana_TSU, kana_TU, kana_TE, kana_TO, kana_NA, kana_NI, kana_NU, kana_NE, kana_NO, kana_HA, kana_HI, kana_FU, kana_HU, kana_HE, kana_HO, kana_MA, kana_MI, kana_MU, kana_ME, kana_MO, kana_YA, kana_YU, kana_YO, kana_RA, kana_RI, kana_RU, kana_RE, kana_RO, kana_WA, kana_N, voicedsound, semivoicedsound, kana_switch,

Arabic_comma, Arabic_semicolon, Arabic_question_mark, Arabic_hamza, Arabic_maddaonalef, Arabic_hamzaonalef, Arabic_hamzaonwaw, Arabic_hamzaunderalef, Arabic_hamzaonyeh, Arabic_alef, Arabic_beh, Arabic_tehmarbuta, Arabic_teh, Arabic_theh, Arabic_jeem, Arabic_hah, Arabic_khah, Arabic_dal, Arabic_thal, Arabic_ra, Arabic_zain, Arabic_seen, Arabic_sheen, Arabic_sad, Arabic_dad, Arabic_tah, Arabic_zah, Arabic_ain, Arabic_ghain, Arabic_tatweel, Arabic_feh, Arabic_qaf, Arabic_kaf, Arabic_lam, Arabic_meem, Arabic_noon, Arabic_ha, Arabic_heh, Arabic_waw, Arabic_alefmaksura, Arabic_yeh, Arabic_fathatan, Arabic_dammatan, Arabic_kasratan, Arabic_fatha, Arabic_damma, Arabic_kasra, Arabic_shadda, Arabic_sukun, Arabic_switch,

Serbian_dje, Macedonia_gje, Cyrillic_io, Ukrainian_ie, Ukranian_je, Macedonia_dse, Ukrainian_i, Ukranian_i, Ukrainian_yi, Ukranian_yi, Cyrillic_je, Serbian_je, Cyrillic_lje, Serbian_lje, Cyrillic_nje, Serbian_nje, Serbian_tshe, Macedonia_kje, Ukrainian_ghe_with_upturn, Byelorussian_shortu, Cyrillic_dzhe, Serbian_dze, numerosign, Serbian_DJE, Macedonia_GJE, Cyrillic_IO, Ukrainian_IE, Ukranian_JE, Macedonia_DSE, Ukrainian_I, Ukranian_I, Ukrainian_YI, Ukranian_YI, Cyrillic_JE, Serbian_JE, Cyrillic_LJE, Serbian_LJE, Cyrillic_NJE, Ser-

bian_NJE, Serbian_TSHE, Macedonia_KJE, Ukrainian_GHE_WITH_UPTURN, Byelorussian_SHORTU, Cyrillic_DZHE, Serbian_DZE, Cyrillic_yu, Cyrillic_a, Cyrillic_be, Cyrillic_tse, Cyrillic_de, Cyrillic_ie, Cyrillic_ef, Cyrillic_ghe, Cyrillic_ha, Cyrillic_i, Cyrillic_shorti, Cyrillic_ka, Cyrillic_el, Cyrillic_em, Cyrillic_en, Cyrillic_o, Cyrillic_pe, Cyrillic_ya, Cyrillic_er, Cyrillic_es, Cyrillic_te, Cyrillic_u, Cyrillic_zhe, Cyrillic_ve, Cyrillic_softsign, Cyrillic_yeru, Cyrillic_ze, Cyrillic_sha, Cyrillic_e, Cyrillic_shcha, Cyrillic_che, Cyrillic_hardsign, Cyrillic_YU, Cyrillic_A, Cyrillic_BE, Cyrillic_TSE, Cyrillic_DE, Cyrillic_IE, Cyrillic_EF, Cyrillic_GHE, Cyrillic_HA, Cyrillic_I, Cyrillic_SHORTI, Cyrillic_KA, Cyrillic_EL, Cyrillic_EM, Cyrillic_EN, Cyrillic_O, Cyrillic_PE, Cyrillic_YA, Cyrillic_ER, Cyrillic_ES, Cyrillic_TE, Cyrillic_U, Cyrillic_ZHE, Cyrillic_VE, Cyrillic_SOFTSIGN, Cyrillic_YERU, Cyrillic_ZE, Cyrillic_SHA, Cyrillic_E, Cyrillic_SHCHA, Cyrillic_CHE, Cyrillic_HARDSIGN,

Greek_ALPHAaccent, Greek_EPSILONaccent, Greek_ETAaccent, Greek_IOTAaccent, Greek_IOTAdiaeresis, Greek_OMICRONaccent, Greek_UPSILONaccent, Greek_UPSILONdieresis, Greek_OMEGAaccent, Greek_accentdieresis, Greek_horizbar, Greek_alphaaccent, Greek_epsilonaccent, Greek_etaaccent, Greek_iotaaccent, Greek_iotadieresis, Greek_iotaaccentdieresis, Greek_omicronaccent, Greek_upsilonaccent, Greek_upsilondieresis, Greek_upsilonaccentdieresis, Greek_omegaaccent, Greek_ALPHA, Greek_BETA, Greek_GAMMA, Greek_DELTA, Greek_EPSILON, Greek_ZETA, Greek_ETA, Greek_THETA, Greek_IOTA, Greek_KAPPA, Greek_LAMDA, Greek_LAMBDA, Greek_MU, Greek_NU, Greek_XI, Greek_OMICRON, Greek_PI, Greek_RHO, Greek_SIGMA, Greek_TAU, Greek_UPSILON, Greek_PHI, Greek_CHI, Greek_PSI, Greek_OMEGA, Greek_alpha, Greek_beta, Greek_gamma, Greek_delta, Greek_epsilon, Greek_zeta, Greek_eta, Greek_theta, Greek_iota, Greek_kappa, Greek_lamda, Greek_lambda, Greek_mu, Greek_nu, Greek_xi, Greek_omicron, Greek_pi, Greek_rho, Greek_sigma, Greek_finalsmallsigma, Greek_tau, Greek_upsilon, Greek_phi, Greek_chi, Greek_psi, Greek_omega, Greek_switch,

hebrew_doublelowline, hebrew_aleph, hebrew_bet, hebrew_beth, hebrew_gimel, hebrew_gimmel, hebrew_dalet, hebrew_daleth, hebrew_he, hebrew_waw, hebrew_zain, hebrew_zayin, hebrew_chet, hebrew_het, hebrew_tet, hebrew_teth, hebrew_yod, hebrew_finalkaph, hebrew_kaph, hebrew_lamed, hebrew_finalmem, hebrew_mem, hebrew_finalnun, hebrew_nun, hebrew_samech, hebrew_samekh, hebrew_ayin, hebrew_finalpe, hebrew_pe, hebrew_finalzade, hebrew_finalzadi, hebrew_zade, hebrew_zadi, hebrew_qoph, hebrew_kuf, hebrew_resh, hebrew_shin, hebrew_taw, hebrew_taf, Hebrew_switch,

Thai_kokai, Thai_khokhai, Thai_khokhuat, Thai_khokhwai, Thai_khokhon, Thai_khorakhang, Thai_ngongu, Thai_chochan, Thai_choching, Thai_chochang, Thai_soso, Thai_chochoe, Thai_yoying, Thai_dochada, Thai_topatak, Thai_thothan, Thai_thonangmontho, Thai_thophuthao, Thai_nonen, Thai_dodek, Thai_totao, Thai_thothung, Thai_thothahan, Thai_thothong, Thai_nonu,

Thai_bobaimai, Thai_popla, Thai_phophung, Thai_fofa, Thai_phophan, Thai_fofan, Thai_phosamphao, Thai_moma, Thai_yoyak, Thai_rorua, Thai_ru, Thai_loling, Thai_lu, Thai_wowaen, Thai_sosala, Thai_sorusi, Thai_sosua, Thai_hohip, Thai_lochula, Thai_oang, Thai_honokhuk, Thai_paiyannoi, Thai_saraa, Thai_maihanakat, Thai_saraaa, Thai_saraam, Thai_sarai, Thai_saraii, Thai_saraue, Thai_sarauee, Thai_sarau, Thai_sarauu, Thai_phinthu, Thai_maihanakat_maitho, Thai_baht, Thai_sarae, Thai_saraae, Thai_sarao, Thai_saraaimaimuan, Thai_saraaimaimalai, Thai_lakkhangyao, Thai_maiyamok, Thai_maitaikhu, Thai_maiek, Thai_maitho, Thai_maitri, Thai_maichattawa, Thai_thanthakhat, Thai_nikhahit, Thai_leksun, Thai_leknung, Thai_leksong, Thai_leksam, Thai_leksi, Thai_lekha, Thai_lekhok, Thai_lekchet, Thai_lekpaet, Thai_lekkao,

Hangul, Hangul_Start, Hangul_End, Hangul_Hanja, Hangul_Jamo, Hangul_Romaja, Hangul_Codeinput, Hangul_Jeonja, Hangul_Banja, Hangul_PreHanja, Hangul_PostHanja, Hangul_SingleCandidate, Hangul_MultipleCandidate, Hangul_PreviousCandidate, Hangul_Special, Hangul_switch, Hangul_Kiyeog, Hangul_SsangKiyeog, Hangul_KiyeogSios, Hangul_Nieun, Hangul_NieunJieuj, Hangul_NieunHieuh, Hangul_Dikeud, Hangul_SsangDikeud, Hangul_Rieul, Hangul_RieulKiyeog, Hangul_RieulMieum, Hangul_RieulPieub, Hangul_RieulSios, Hangul_RieulTieut, Hangul_RieulPhieuf, Hangul_RieulHieuh, Hangul_Mieum, Hangul_Pieub, Hangul_SsangPieub, Hangul_PieubSios, Hangul_Sios, Hangul_SsangSios, Hangul_Ieung, Hangul_Jieuj, Hangul_SsangJieuj, Hangul_Cieuc, Hangul_Khieuq, Hangul_Tieut, Hangul_Phieuf, Hangul_Hieuh, Hangul_A, Hangul_AE, Hangul_YA, Hangul_YAE, Hangul_EO, Hangul_E, Hangul_YEO, Hangul_YE, Hangul_O, Hangul_WA, Hangul_WAE, Hangul_OE, Hangul_YO, Hangul_U, Hangul_WEO, Hangul_WE, Hangul_WI, Hangul_YU, Hangul_EU, Hangul_YI, Hangul_I, Hangul_J_Kiyeog, Hangul_J_SsangKiyeog, Hangul_J_KiyeogSios, Hangul_J_Nieun, Hangul_J_NieunJieuj, Hangul_J_NieunHieuh, Hangul_J_Dikeud, Hangul_J_Rieul, Hangul_J_RieulKiyeog, Hangul_J_RieulMieum, Hangul_J_RieulPieub, Hangul_J_RieulSios, Hangul_J_RieulTieut, Hangul_J_RieulPhieuf, Hangul_J_RieulHieuh, Hangul_J_Mieum, Hangul_J_Pieub, Hangul_J_PieubSios, Hangul_J_Sios, Hangul_J_SsangSios, Hangul_J_Ieung, Hangul_J_Jieuj, Hangul_J_Cieuc, Hangul_J_Khieuq, Hangul_J_Tieut, Hangul_J_Phieuf, Hangul_J_Hieuh, Hangul_RieulYeorinHieuh, Hangul_SunkyeongeumMieum, Hangul_SunkyeongeumPieub, Hangul_PanSios, Hangul_KkogjiDalrinIeung, Hangul_SunkyeongeumPhieuf, Hangul_YeorinHieuh, Hangul_AraeA, Hangul_AraeAE, Hangul_J_PanSios, Hangul_J_KkogjiDalrinIeung, Hangul_J_YeorinHieuh, Korean_Won,

Armenian_eternity, Armenian_section_sign, Armenian_full_stop, Armenian_verjaket, Armenian_parenright, Armenian_parenleft, Armenian_guillemotright, Armenian_guillemotleft, Armenian_em_dash, Armenian_dot, Armenian_mijaket, Armenian_separation_mark, Armenian_but, Armenian_comma, Armenian_en_dash, Armenian_hyphen, Armenian_yentamna, Armenian_ellipsis, Arme-

nian_exclam, Armenian_amanak, Armenian_accent, Armenian_shesht, Armenian_question, Armenian_paruyk, Armenian_AYB, Armenian_ayb, Armenian_BEN, Armenian_ben, Armenian_GIM, Armenian_gim, Armenian_DA, Armenian_da, Armenian_YECH, Armenian_yech, Armenian_ZA, Armenian_za, Armenian_E, Armenian_e, Armenian_AT, Armenian_at, Armenian_TO, Armenian_to, Armenian_ZHE, Armenian_zhe, Armenian_INI, Armenian_ini, Armenian_LYUN, Armenian_lyun, Armenian_KHE, Armenian_khe, Armenian_TSA, Armenian_tsa, Armenian_KEN, Armenian_ken, Armenian_HO, Armenian_ho, Armenian_DZA, Armenian_dza, Armenian_GHAT, Armenian_ghat, Armenian_TCHE, Armenian_tche, Armenian_MEN, Armenian_men, Armenian_HI, Armenian_hi, Armenian_NU, Armenian_nu, Armenian_SHA, Armenian_sha, Armenian_VO, Armenian_vo, Armenian_CHA, Armenian_cha, Armenian_PE, Armenian_pe, Armenian_JE, Armenian_je, Armenian_RA, Armenian_ra, Armenian_SE, Armenian_se, Armenian_VEV, Armenian_vev, Armenian_TYUN, Armenian_tyun, Armenian_RE, Armenian_re, Armenian_TSO, Armenian_tso, Armenian_VYUN, Armenian_vyun, Armenian_PYUR, Armenian_pyur, Armenian_KE, Armenian_ke, Armenian_O, Armenian_o, Armenian_FE, Armenian_fe, Armenian_apostrophe, Armenian_ligature_ew,

Georgian_an, Georgian_ban, Georgian_gan, Georgian_don, Georgian_en, Georgian_vin, Georgian_zen, Georgian_tan, Georgian_in, Georgian_kan, Georgian_las, Georgian_man, Georgian_nar, Georgian_on, Georgian_par, Georgian_zhar, Georgian_rae, Georgian_san, Georgian_tar, Georgian_un, Georgian_phar, Georgian_khar, Georgian_ghan, Georgian_qar, Georgian_shin, Georgian_chin, Georgian_can, Georgian_jil, Georgian_cil, Georgian_char, Georgian_xan, Georgian_jhan, Georgian_hae, Georgian_he, Georgian_hie, Georgian_we, Georgian_har, Georgian_hoe, Georgian_fi,

## 2.5. Datei-Sets

Wing stellt eine Möglichkeit zur Definition von Datei-Sets bereit, die auf verschiedene Weise innerhalb des IDEs verwendet werden können, zum Beispiel für das Durchsuchen bestimmter Dateistapel und für das Hinzufügen nur bestimmter Dateiarten zu einem Projekt.

Verwenden Sie den Eintrag `Datei-Sets...` aus dem Menü Datei, um die definierten Datei-Sets anzuzeigen oder zu ändern. Dies wird innerhalb des Einstellungsmanagers einen Datei-Set-Editor anzeigen.

Wenn Sie ein Datei-Set hinzufügen oder bearbeiten, können Sie die folgenden Informationen eingeben:

- **Name** -- Der Name des Datei-Sets.

- **Einschließen** -- Eine Liste von Einschlusskriterien, von denen jedes eine Art und eine Spezifizierung enthält. Eine Datei wird in das Datei-Set einbezogen, wenn irgendeins dieser Einschlusskriterien zutrifft.
- **Ausschließen** -- Eine Liste von Ausschlusskriterien, von denen jedes beliebige zutreffen kann, was dann verursacht, dass eine Datei von dem Datei-Set ausgeschlossen wird, selbst wenn auch ein oder mehrere Einschlussübereinstimmungen gefunden werden.

Die folgenden Arten von Einschluss- und Ausschlusskriterien werden unterstützt:

- **Dateiname-Wildcard** -- Die Spezifikation ist in diesem Fall eine Wildcard, die mit dem Dateinamen übereinstimmen muss. Die unterstützten Wildcards sind diejenigen, die von Python's fnmatch Modul bereitgestellt werden.
- **Mime-Typ** -- Die Spezifikation benennt in diesem Fall einen MIME-Typen, der von Wing IDE unterstützt wird. Wenn für diese Art der Spezifikation zusätzliche MIME-Typen benötigt werden, denn verwenden Sie die Einstellung **Extra-Mime-Typen**, um sie zu definieren.

Sobald sie definiert sind, werden Datei-Sets nach Namen in der Batch-Sucheinrichtung des **Suchen/Ersetzen** Werkzeuges und in den Batch-Dateizusatzfunktionen des **Projekt** Werkzeuges dargestellt.

Probleme, die bei der Verwendung des Datei-Sets auftreten, werden im Bereich Nachrichten berichtet.

# Projektmanager

Der Projektmanager stellt einen praktischen Index der Dateien in Ihrem Software-Projekt bereit und sammelt Informationen, die von Wing's Debugger, vom Werkzeug für die Source-Code-Analyse und anderen Einrichtungen benötigt werden.

Um das Beste aus Wing's Debugger und der Source-Analyse-Maschine herauszuholen, müssen Sie Ihre Source-Basis zu Ihren Projektdateien hinzufügen und werden in einigen Fällen `PYTHONPATH` und andere Werte in den **Projektweiten Eigenschaften** und/oder **Pro-Datei Eigenschaften** einrichten müssen.

## 3.1. Ein Projekt erstellen

Verwenden Sie den Eintrag `Neues Projekt` aus dem Projektmenü, um ein neues Projekt zu erstellen. Dies wird Sie dazu auffordern, alle Änderungen in Ihrem gegenwärtig geöffneten Projekt zu speichern und wird ein neues, leeres Projekt erstellen. Wenn Sie Wing ohne Argumente in der Befehlszeile starten, dann wird standardmäßig ein neues, leeres Projekt geöffnet.

Um Dateien zu Ihrem Projekt hinzuzufügen, verwenden Sie die folgenden Einträge aus dem Projektmenü:

- **Aktuelle Datei hinzufügen** wird die vorderste der gegenwärtig geöffneten Dateien zum Projekt hinzufügen, wenn diese nicht bereits dort vorhanden ist.
- **Datei hinzufügen** wird Sie auffordern, eine einzelne Datei zur Projektansicht hinzuzufügen. Beachten Sie, dass dies auch zum Hinzufügen eines neuen Verzeichnisses zum Projektmanagerfenster führen kann, wenn die Datei die erste ist, die in ein Verzeichnis hinzugefügt wird.
- **Paket hinzufügen** kann verwendet werden, um mehr als eine Datei gleichzeitig hinzuzufügen. Wählen Sie ein Verzeichnis mit Ihrer linken Maustaste, so dass der Verzeichnisname im unteren Bereich des Dateiauswahlfensters angezeigt wird. Klicken Sie dann auf OK. Ihnen wird eine Liste von Dateien innerhalb des gewählten

Verzeichnisses angezeigt werden. Markieren Sie alle, die Sie hinzufügen möchten. Verwenden Sie Umschalt-Klick, um einen zusammenhängenden Bereich zu markieren oder Strg-Klick, um eine beliebige Auswahl zu treffen. Sie können auch aus dem Popup-Menü `Filter` auswählen, um vordefinierte Dateisets anzugeben, um die Liste der Dateien im Paket zu filtern. Klicken Sie dann auf Ja, um alle diese Dateien zu Ihrem Projekt hinzuzufügen.

- **Verzeichnisbaum hinzufügen** kann verwendet werden, um viele Dateien in eine Verzeichnisstruktur mit einem Arbeitsvorgang hinzuzufügen. Wählen Sie ein Verzeichnis aus der bereitgestellten Liste. Optional: Wählen Sie aus dem Popup-Menü `Filter`, um vordefinierte Dateisets anzugeben, um die Liste der Dateien, die hinzugefügt werden, zu filtern. Sobald dieser Dialog akzeptiert wurde, wird Wing die Dateien rekursiv aus dem gewählten Verzeichnis und allen seinen Kindern hinzufügen.

Diese Optionen können Sie auch über das Popup-Menü erreichen, das erscheint, wenn Sie mit der rechten Maustaste auf die Oberfläche des Projektmanagerfensters klicken.

## 3.2. Dateien und Pakete entfernen

Um eine spezifische Datei zu entfernen, markieren Sie diese und verwenden den Menüeintrag `Vom Projekt entfernen` aus dem Popup-Menü, das mit einem rechten Mausklick auf die Oberfläche des Projektmanagerfensters erreicht wird. Eine andere Möglichkeit zum Entfernen besteht darin, einen Eintrag aus dem Projekt zu markieren und den Punkt `Markierten Eintrag entfernen` aus dem Projektmenü zu verwenden. Sie können auch ein ganzes Verzeichnis und alle Dateien, die es enthält, auf diese Weise entfernen.

## 3.3. Das Projekt speichern

Sobald eine Projektdatei das erste Mal gespeichert wurde, wird sie automatisch immer wieder gespeichert, wenn Sie das Projekt schließen, eine Debug-Sitzung starten oder Wing beenden. Dieses Verhalten kann mit der Einstellung **Speichern ohne zu Fragen** ausgeschalten werden.

Sie können auch eine Kopie von Ihrem Projekt an einem anderen Ort oder mit einem anderen Namen speichern, wenn Sie den Eintrag `Projekt speichern unter...` aus dem Projektmenü verwenden.

**Projektdateien verschieben**
Die Verwendung von `Projekt speichern unter...` wird empfohlen, wenn Sie den Ort Ihrer Projektdatei in Bezug auf Ihre Source-Dateien verändern müssen, weil es die teilweise relativen Pfade, die der Projektmanager verwendet, um Dateien im Projekt zu lokalisieren, aktualisiert. Andernfalls kann Wing nicht in der Lage sein, alle Dateien im Projekt zu finden.

## 3.4. Die Ansicht sortieren

Das Projekt kann so eingestellt werden, dass es Ihre Dateien in verschiedenen Modi anzeigt. Verwenden Sie dafür das Menü `Optionen` in der oberen rechten Ecke der Projektansicht:

- **Nach abgeflachtem Baum** -- Diese Ansicht (Voreinstellung) zeigt die Dateien geordnet nach ihrem Ort auf dem Laufwerk an. Jedes Verzeichnis wird in der höchsten Ebene angezeigt, und zwar mit Pfadnamen, die als teilweise relative Pfade basierend auf dem Ort der Projektdatei angezeigt werden. Wenn Sie den Ort der Projektdatei mit `Projekt speichern unter...` ändern, werden diese Pfade entsprechend aktualisiert.
- **Nach Baum** -- Dies zeigt die Projektdateien in echter Baumform an. Die Baumstruktur basiert auf dem teilweise relativen Pfad von der Projektdatei.
- **Nach Mime-Typ** -- Diese Ansicht ordnet Ihre Dateien nach MIME-Typ.

## 3.5. Tastaturnavigation

Sobald sie den Fokus eingestellt hat, ist die Baumansicht des Projektmanagers mit der Tastatur steuerbar, indem Sie die Pfeiltasten oben/unten, Bild oben und Bild unten sowie Pos1/Ende verwenden.

Verwenden Sie die rechte Pfeiltaste auf einem Parent, um dessen Abkömmlinge anzuzeigen oder die linke Pfeiltaste, um sie zu verstecken.

Wenn Sie die Umschalttaste gedrückt halten, während Sie die rechte Pfeiltaste drücken, wird unter dem Erweiterungspunkt rekursiv erweitert. Die rekursive Erweiterung ist auf fünf zusätzlichen Ebenen für jede Operation begrenzt, um die unendliche Rekursion, die aus symbolischen Links resultiert, zu vermeiden.

Immer wenn eine Baumreihe markiert ist, wird das Drücken der Eingabe- oder Return-Taste das Objekt in Wing IDE öffnen.

## 3.6. Gemeinsame Nutzung von Projekten

> **Arten von Projektdateien**
> Es gibt zwei verwandte Formate, in denen Sie Ihr Projekt speichern können. Eines unterstützt die gemeinsame Nutzung von Projektdateien mit anderen Entwicklern über ein Revisionskontrollsystem oder einer anderen Methode.

Der Standard-Projekttyp ist 'Normal', wobei alle Projektdaten in einer einzelnen Datei gespeichert werden. Der Dateiname für diese Dateien sollte mit `'.wpr'` enden.

Um ein Projekt mit anderen Entwicklern gemeinsam zu nutzen, ändern Sie die `Projektart` im Reiter `Optionen` der `Projekteigenschaften` auf " Gemeinsam (Zwei Dateien)". Speichern Sie dann Ihr Projekt, um zwei separate Projektdateien auf dem Laufwerk zu erhalten. Der Hauptprojekt-Dateiname endet auf `'.wpr'` und wird nur gemeinsam nutzbare Daten enthalten. Alle nutzerspezifischen Daten werden in einer separaten Datei gespeichert, deren Name auf `'.wpu'` endet.

Wenn Sie später vom einem gemeinsamen Projekt zurück zu Normal wechseln, werden die nutzerspezifischen Daten wieder mit der Projektdatei auf der Maschine zusammengefügt und die Datei, die auf `'.wpu'` endet, wird vom Laufwerk entfernt werden.

Beachten Sie, dass sowohl die kombinierte 'Normal'-Datei als auch die zwei geteilten 'gemeinsamen' Dateien das gleiche Textdateiformat verwenden, das für die Einstellungsdatei verwendet wird. Lesen Sie den Abschnitt **Format der Einstellungsdatei** für weitere Informationen über das Format selbst.

## 3.7. Projektweite Eigenschaften

Jedes Projekt hat ein Set von Top-Level Eigenschaften, die über den Eintrag **Eigenschaften** im Projektmenü erreicht und bearbeitet werden können. Sie können verwendet werden, um die Python-Umgebung zu konfigurieren. Die Python-Umgebung wird vom **Debugger** und der Maschine für die **Source-Code-Analyse**, welche Wing's Auto-Vervollständigung, Source-Index und andere Funktionen betreiben, verwendet. Projekteinstellungen werden außerdem bereitgestellt, um Optionen für das Projekt einzustellen und um Erweiterungen für die Revisionskontrolle, Zope und andere Werkzeuge zu aktivieren und konfigurieren.

Jeder String-Wert für eine Eigenschaft kann Verweise zu Umgebungsvariablen enthalten und dafür die `$(name)` Notation verwenden. Alles innerhalb der Klammern wird als Name einer Umbgebungsvariablen interpretiert und wird mit dem Wert der Umgebungsvariablen ersetzt, wenn es vom IDE verwendet wird. Wenn die Umgebungsvariable nicht

gesetzt ist, wird der Verweis mit einem leeren String ersetzt. Die System-Umgebung, so wie von der projektweiten Umgebungseigenschaft geändert (siehe unten), wird verwendet, um Variablenverweise zu erweitern.

- **Python-Einstellungen***

Um das Beste aus Wing herauszuholen, ist es wichtig, dass Sie diese Werte im Reiter Python-Einstellungen korrekt für Ihr Projekt einstellen:

**Python-Executable** -- Wenn die Option *Benutzerdefinierte Einstellung* markiert ist und das eingegebene Feld nicht-leer ist, kann dies verwendet werden, um den vollen Pfad zur Python-Executable einzustellen, die verwendet werden sollte, wenn Source-Code im Projekt gedebuggt wird. Wenn *Standard verwenden* markiert ist, versucht Wing, das Standard-Python zu verwenden, das erhalten werden kann, wenn `python` in der Command Line eingegeben wird. Wenn dies scheitert, wird Wing nach Python in `/usr/local` und `/usr` (in Linux/Unix) oder in der Registratur (in Windows) suchen.

**Python-Pfad** -- Der `PYTHONPATH` wird von Python verwendet, um Module zu lokalisieren, die während der Laufzeit mit der `import` Anweisung importiert werden. Wenn das Kontrollkästchen *Standard verwenden* in diesem Bereich markiert ist, wird die geerbte `PYTHONPATH` Umgebungsvariable für Debug-Sitzungen verwendet. Wenn dagegen *Benutzerdefinierte Einstellung* gewählt ist, wird der angegebene `PYTHONPATH` verwendet.

- **Debug-Einstellungen**

Die folgenden Einstellungen sind im Debug-Reiter definiert:

**Startverzeichnis** -- Wenn die Option *Standard verwenden* markiert ist, wird das anfängliche Arbeitsverzeichnis, das für jede Debug-Sitzung eingestellt ist, der Ort sein, an dem die Datei des Debug-Startpunktes platziert ist. Wenn dagegen *Benutzerdefinierte Einstellung* gewählt ist, wird das angegebene Verzeichnis verwendet oder, wenn dies leer ist, wird das Verzeichnis der Projektdatei genutzt.

**Build-Befehl** -- Dieser Befehl wird ausgeführt, bevor das Debuggen von Source-Code in diesem Projekt begonnen wird. Dies ist hilfreich, um sicherzustellen, dass C/C++ Erweiterungsmodule erstellt werden, zum Beispiel in Verbindung mit einem externen `Makefile` oder `distutils`-Skript, bevor die Ausführung gestartet wird.

**Umgebung** -- Dies wird verwendet, um Werte zu bestimmen, die zu der Umgebung, welche von den von Wing IDE gestarteten Debug-Prozessen geerbt ist, hinzugefügt, geändert oder entfernt werden sollen. Dies wird auch verwendet, um Verweise zu Umgebungsvariablen, die in anderen Eigenschaften bestimmt sind, zu erweitern. Jeder Eintrag ist in der Form `var=value` und muss auf einer eigenen Zeile in dem bereitgestellten Eingabebereich angegeben werden. Ein Eintrag in der Form `var=` (ohne einen Wert)

entfernt die gegebene Variable, so dass sie undefiniert ist. Beachten Sie, dass Sie in der Umgebung arbeiten, die vom IDE geerbt wurde, als es gestartet wurde, und dass Sie keine leere Umgebung verändern. Wenn die Option *System-Umgebung verwenden* gewählt ist, werden alle eingegebenen Werte ignoriert und die geerbte Umgebung wird ohne Änderungen verwendet.

- **Projektoptionen**

Die folgenden Projektoptionen stehen zur Verfügung:

**Projektart** -- Mit dieser Eigenschaft können Sie bestimmen, ob ein Projekt von mehreren Entwicklern gemeinsam genutzt werden soll. Bei einer gemeinsamen Nutzung wird das Projekt in zwei Dateien gespeichert, von denen eine zur gemeinsamen Verwendung mit anderen Entwicklern bereit steht. Siehe **Projektarten** für Einzelheiten.

**Bevorzugtes Zeilenende** und **Zeilenendengrundsatz** steuern, ob für das Projekt eine bestimmte Art für das Zeilenende (Zeilenvorschub (LF), Carriage Return (CR, Cursor kehrt zum Zeilenanfang zurück) oder Carriage Return und Zeilenvorschub (CRLF)) bevorzugt wird. Außerdem bestimmen die Eigenschaften, wie die entsprechende Art durchgeführt wird, wenn zutreffend. Standardmäßig erzwingen Projekte keine Art für das Zeilenende, sondern fügen stattdessen neue Zeilen ein, um mit den vorhandenen Zeilenenden in der Datei übereinzustimmen.

**Bevorzugter Einrückungsstil** und **Einrückungsgrundsatz** steuern, ob für das Projekt eine bestimmte Einrückungsart (Nur Leerzeichen, Nur Tabs, Gemischte Tabs und Leerzeichen) für die Dateien bevorzugt werden soll und wie die entsprechende Art durchgeführt wird, wenn zutreffend. Standardmäßig erzwingen Projekte keinen bestimmten Einrückungsstil, sondern fügen stattdessen neue Zeilen ein, um mit den vorhandenen Einrückungen in der Datei übereinzustimmen.

- **Erweiterungen**

Der Reiter Erweiterungen in den Projekteigenschaften wird zur Steuerung der Revisionskontrolle und anderer Add-ons auf einer Pro-Projekt Basis verwendet:

**Revisionskontrolle aktivieren** und **Revisionskontrollsystem** werden verwendet, um für dieses Projekt die Integration einer bestimmten Revisionskontrolle zu aktivieren. Zur Zeit ist nur eine minimale CVS-Integration verfügbar.

**Zope/Plone-Support aktivieren** und **Home der Zope/Plone-Instanz** werden für Zope 2.x und Plone-Projekte verwendet, um das von Zope verwendete Home-Verzeichnis der Instanz bereitzustellen. Dies ist notwendig, da Zope 2.x „Import-Magie“ implementiert, die anders als Python's Standard `import` funktioniert und es daher nicht ausreicht, das Home-Verzeichnis der Instanz zu `PYTHONPATH` hinzuzufügen. Wing's Source-Analyser

braucht diesen extra Hinweis, um die Zope instanzspezifischen Source-Dateien richtig zu finden und zu verarbeiten.

Wenn Sie eine Projektdatei mit anderen Entwicklern über ein Revisionskontrollsystem gemeinsam nutzen und den Projekttyp auf `Gemeinsam` gesetzt haben, ist es wichtig zu beachten, dass die oben genannten Werte im privaten Zweig der Projektdatei gespeichert werden. Das heißt, dass alle Entwickler diese Werte unabhängig voneinander einstellen müssen, damit diese mit der spezifischen Umgebung auf der jeweiligen Entwicklungsmaschine übereinstimmen.

## 3.8. Pro-Datei Eigenschaften

Eigenschaften auf einer Pro-Datei Basis können auf verschiedene Weisen eingestellt werden: Mit der rechten Maustaste auf eine Source-Datei klicken und aus dem Popup-Menü den Eintrag `Eigenschaften` auswählen oder mit der rechten Maustaste auf eine Datei in der Projektansicht klicken und `Dateieigenschaften` auswählen oder eine Datei öffnen und den Eintrag `Aktuelle Dateieigenschaften...` aus dem Source-Menü verwenden.

- **Dateiattribute**

**Kodierung** -- Dies kann verwendet werden, um die Kodierung, mit der eine Datei gespeichert wird, zu bestimmen. Bei Änderung dieses Wertes wird die Datei in einem Editor geöffnet und die Kodierung wird erst geändert, wenn die Datei auf dem Laufwerk gespeichert wird. Wird sie nicht gespeichert, wird die Kodierung wieder auf die vorherige Einstellung zurückgesetzt. Die Kodierung kann mit dieser Eigenschaft nicht geändert werden, wenn sie mit einem Kodierungskommentar in einer Python, HTML, XML oder gettext PO Datei definiert ist. In diesem Fall sollte die Datei geöffnet und der Kodierungskommentar geändert werden. Wing speichert dann die Datei mit der neu festgelegten Kodierung.

**Wichtig:** Dateien, die mit einer anderen Kodierung ohne Kodierungskommentar gespeichert werden, können unter Umständen von anderen Editoren nicht gelesen werden, weil es keine Möglichkeit gibt, die Kodierung einer Datei zu bestimmen, wenn sie vom System- oder Laufwerkstandard abweicht. Wing speichert die gewählte Kodierung in der Projektdatei, aber in der Datei selbst wird nichts vermerkt, außer für solche Kodierungen, die normalerweise ein Byte Order Mark (BOM) verwenden, wie utf_8, utf_16_le, utf_16_be, utf_32_le, utf_32_be.

**Art des Zeilenendes** -- Bestimmt, welche Art von Zeilenende in der Datei verwendet wird (Zeilenvorschub (LF), Carriage Return (CR, Cursor kehrt zum Zeilenanfang zurück) oder Carriage Return und Zeilenvorschub (CRLF)). Bei Änderung dieser Einstellung

wird die Datei in einem Editor geöffnet und geändert. Die Änderungen werden erst wirksam, wenn die Datei auf dem Laufwerk gespeichert wurde.

**Einrückungsstil** -- Diese Eigenschaft kann für nicht-Python-Dateien verwendet werden, um die Art der Einrückung für neu hinzugefügte Zeilen in der Datei zu bestimmen. Für Python-Dateien können die Einrückungen in einer Datei nur mit dem **Einrückungsmanager** geändert werden.

**Nur Lesen auf dem Laufwerk** -- Diese Eigenschaft zeigt an, ob die Datei auf dem Laufwerk als „Nur Lesen" markiert ist. Die Änderung dieser Eigenschaft ändert den Schutz der Datei auf dem Laufwerk für den Besitzer der Datei (in Posix werden group/world Berechtigungen nie geändert).

- **Editor**

Diese Einstellungen definieren, wie eine Datei im Editor angezeigt wird:

**Syntax-Markierung** -- Diese Eigenschaft bestimmt die Dateiart einer bestimmten Datei und überschreibt die Art, die automatisch vom Dateizusatz und/oder Inhalt ermittelt wird. Diese Einstellung ist nur empfehlenswert, wenn die Einstellung **Extra-Dateiarten** nicht zur Bestimmung der Kodierung basierend auf Dateizusätzen verwendet werden kann.

**Leerraum anzeigen** -- Mit dieser Option kann die Einstellung **Leerraum anzeigen** auf einer Pro-Datei Basis außer Kraft gesetzt werden.

**Zeilenende anzeigen** -- Mit dieser Option kann die Einstellung **Zeilenende anzeigen** auf einer Pro-Datei Basis außer Kraft gesetzt werden.

**Einrückungslinien anzeigen** -- Mit dieser Option kann die Einstellung **Einrückungslinien anzeigen** auf einer Pro-Datei Basis außer Kraft gesetzt werden.

**Einrückungsfehler ignorieren** -- Normalerweise berichtet Wing mögliche schwerwiegende Einrückungsinkonsistenzen in Python-Dateien. Diese Einstellung kann verwendet werden, um diese Prüfung auf einer Pro-Datei Basis zu deaktivieren (sie ist auch im Dialog Warnung verfügbar).

**Zeilenendefehler ignorieren** -- Wenn die Projekteinstellung **Zeilenendegrundsatz** gesetzt ist, um über nicht übereinstimmende Zeilenenden zu warnen, kann diese Einstellung verwendet werden, um die Warnungen für eine bestimmte Datei zu deaktivieren.

- **Debuggen**

Der Dialog für die Debug-Eigenschaften pro Datei enthält die gleichen Felder, die im Abschnitt **Projektweite Eigenschaften** beschrieben sind, mit den folgenden Zusätzen:

**Ausführungsargumente** -- Geben Sie ein beliebiges Ausführungsargument ein. Wing interpretiert Backslashes (") in der Command Line nicht und gibt diese unverändert an den Debug-Prozess weiter. Die einzige Ausnahme dieser Regel sind `\'` und `\"` (Backslash gefolgt von einfachen oder doppelten Anführungszeichen), die die Einbeziehung von Anführungszeichen innerhalb von zitierten Argumenten, die mehrere Wörter umfassen, erlauben.

- **Umgebung** -- Der Menübereich Optionen enthält ein paar zusätzliche

Bullet list ends without a blank line; unexpected unindent.

Auswahlmöglichkeiten. Verwenden Sie *Zu Projektwerten hinzufügen*, um die hier bestimmten Werte auf die vom Projekt bestimmte Ausführungsumgebung anzuwenden. Sie können auch *Zur Systemumgebung hinzufügen* nutzen, um die projektweiten Werte zu umgehen und die pro-Datei-Werte direkt auf die vom Betriebssystem gesetzte Umgebung anzuwenden.

- **Diesen Dialog vor jedem Durchlauf anzeigen** -- Markieren Sie dieses

Bullet list ends without a blank line; unexpected unindent.

Kontrollkästchen, wenn Sie möchten, dass der Dialog der Debug-Optionen jedesmal, wenn Sie eine Debug-Sitzung starten, erscheint.

Werte, die für eine Datei festgelegt werden, setzen die entsprechende projektweite Einstellung außer Kraft oder ändern diese.

Wenn Sie debuggen, werden nur die Pro-Datei Debug-Eigenschaften, die in der *anfänglich aufgerufenen Datei* eingestellt sind, verwendet. Selbst wenn andere Dateien mit eingerichteten Eigenschaften in der Debug-Sitzung verwendet werden, werden alle für sie eingestellten Werte ignoriert.

- **Python-Einstellungen**

Diese Einstellungen sind die gleichen, wie die in den **Projektweiten Einstellungen** definierten Python-Einstellungen. Werte, die für eine Datei festgelegt werden, setzen die entsprechende projektweite Einstellung außer Kraft.

## 3.9. Dateiinformationen anzeigen

Der untere Teil des Projektmanagerfensters enthält einen Bereich mit Dateiinformationen, der den Dateinamen, Dateityp und den Dokumentationsstring für die Dateien (wenn

verfügbar) für die aktuelle Auswahl im Dateiauswahlbereich des Projektmanagerfensters anzeigt.

Der Dokumentationsstring enthält den Docstring der Dateiebene nur für Python-Dateien und unterstützt zur Zeit keine anderen Programmiersprachen.

Die Größe dieses Bereiches kann verändert werden, indem Sie die Teilungslinie zwischen dem Dateiinformationsbereich und dem Rest des Projektmanagerfensters verschieben.

## 3.10. Navigation zu Dateien

Dateien können vom Projektmanagerfenster geöffnet werden, indem Sie entweder doppelt oder mit der mittleren Maustaste auf den Dateinamen klicken oder Sie können mit der rechten Maustaste klicken und den Menüpunkt `In Wing IDE öffnen` verwenden.

Dateien können auch unter Verwendung einer externen Ansicht oder eines Editors geöffnet werden, indem Sie mit der rechten Maustaste auf die Datei klicken und den Eintrag `In externer Ansicht öffnen` verwenden. In Windows und Mac OS X öffnet dies die Datei, so als ob Sie sie doppelt angeklickt hätten. In Linux können Sie die Einstellungen **Befehle der Dateianzeige** und **Extra-Mime-Typen** verwendet, um zu konfigurieren, wie Dateien geöffnet werden.

Sie können Makefiles, Python-Source und alle ausführbaren Dateien auch ausführen, indem Sie `Gewählte Datei ausführen` aus dem Popup-Menü wählen. Dies führt außerhalb des Debuggers aus, mit aller Eingabe/Ausgabe, die in dem Fenster, von dem Wing gestartet wurde (wenn vorhanden), auftritt.

# Source-Code-Editor

Wing IDE's Source-Code-Editor ist so gestaltet, dass es einfach für Sie ist, mit dem IDE zu arbeiten, selbst wenn Sie an andere Editoren gewöhnt sind.

**Editor-Übersicht**
Schlüsselelemente, die Sie über den Editor wissen sollten:

- Der Editor hat Individualitäten, einschließlich einer, die Standard-Editoren in Windows ähnlich ist und einer anderen ähnlich zu Emacs.
- Tastaturkombinationen sind konfigurierbar.
- Der Editor unterstützt Syntax-Farbmarkierungen für eine breite Auswahl von Dateitypen.
- Der Editor unterstützt strukturelles Falten für einige Dateitypen.
- Auto-Vervollständigung wird für Python-Source unterstützt.

## 4.1. Syntax-Farbmarkierung

Der Editor wird versuchen, Dokumente entsprechend ihres MIME-Typen, welcher vom Dateizusatz bestimmt wird, oder entsprechend ihres Inhalts zu markieren. Zum Beispiel wird jede Datei, die mit".py" endet, als ein Python Source-Code-Dokument markiert. Jede Datei, deren MIME-Typ nicht bestimmt werden kann, wird den gesamten Text standardmäßig in schwarzer Normalschrift anzeigen.

Alle verfügbaren Dokumenttypen für Farbmarkierungen sind im Dialog Dateieigenschaften im Editor-Reiter aufgelistet. Wenn Sie mit einer Datei arbeiten, die nicht automatisch erkannt wird, können Sie das Menü Syntax-Markierung verwenden, um die Art, wie diese Datei angezeigt wird, zu ändern. Die Auswahl aus diesem Menü wird in Ih-

rer Projektdatei gespeichert, so dass hier vorgenommene Änderungen im Kontext dieses Projektes dauerhaft sind.

Wenn Sie viele Dateien mit einer unerkannten Erweiterung haben, verwenden Sie die Einstellung **Extra-Mime-Typen**, um Ihre Erweiterung hinzuzufügen.

## 4.2. Rechtsklick-Menü des Editors

**Popup-Menü des Editors**
Ein rechter Mausklick auf die Oberfläche des Editors schlägt ein Popup-Menü mit allgemein verwendeten Befehlen, wie Kopieren, Einfügen, Rückgängig und Wiederherstellen, auf. Wenn die Datei eine Python-Datei ist, enthält dieses Menü auch einen Befehl, um zum Punkt der Definition für den Wert, auf den geklickt wurde, zu zoomen.

## 4.3. Source-Code-Navigation

Das Set von Menüs am Anfang des Editors kann verwendet werden, um durch Ihren Source-Code zu navigieren. Wenn die Reiter des Editor Notizbuches unsichtbar sind, enthaltet die am weitesten links gelegenen Menü eine Liste des geöffneten Dateis. Die zusätzlichen Popup-Menüs zeigen den Bereich der aktuellen Cursor-Auswahl in der Datei an und können verwendet werden, um innerhalb des Top-Level-Bereichs oder innerhalb von Unterbereichen, wenn diese existieren, zu navigieren.

Sie können auch den Menüpunkt `Gehe zur Definition` aus dem Popup-Menü, das mit mit einem rechten Mausklick aufgeschlagen wird, verwenden, um auf ein Konstrukt in Ihrem Source-Code zu klicken und zu dessen Punkt der Definition zu zoomen. Alternativ können Sie den Cursor oder die Auswahl auf einem Symbol platzieren und den Punkt `Gehe zur gewählten Symboldefiniton` aus dem Menü Source oder die entsprechende Tastaturkombination verwenden.

## 4.4. Dateistatus und nur lesbare Dateien

Die Editor-Reiter oder das Auswahlmenü des Editors (wenn die Reiter versteckt sind) zeigen den Status einer Datei an. Es wird ein `*` angehängt, wenn die Datei bearbeitet wurde, oder `(r/o)` (read-only) hinzugefügt, wenn die Datei nur lesbar ist und nicht geändert werden kann. Diese Information wird für die aktuelle Datei im Statusbereich in der unteren linken Ecke jedes Editor-Fensters gespiegelt.

Dateien, die auf dem Laufwerk nur lesbar sind, werden anfangs in einem nur lesbaren Editor geöffnet. Verwenden Sie das Kontextmenü der Datei (rechter Mausklick), um zwischen dem nur lesbaren Zustand und dem beschreibbaren Zustand zu wechseln. Dies ändert nur die Editierbarkeit des Editors und versucht nicht, den Status der Datei (nur lesbar oder beschreibbar) zu ändern.

## 4.5. Vorübergehende vs. nicht vorübergehende Editoren

Wing kann Dateien in zwei Modi öffnen:

**Modus „Vorübergehend“** -- Dateien, die beim Suchen, Debuggen, Navigieren zum Punkt der Definition sowie bei der Verwendung der Werkzeuge Projekt und Source-Browser (mit dem Kontrollkästchen `Auswahl folgen` aktiviert), geöffnet werden, werden immer im Modus „Vorübergehend“ geöffnet. Diese Dateien werden automatisch geschlossen, wenn sie versteckt werden. Die maximale Anzahl nicht-sichtbarer, vorübergehender Dateien, die jederzeit geöffnet bleiben, kann mit der Einstellung **Editor / Erweitert / Schwelle für vorübergehende Dateien** festgelegt werden.

**Modus „Nicht vorübergehend“** -- Dateien, die über das Menü Datei, mit der Dateiauswahl über die Tastatur oder mit einem Doppelklick auf Einträge im Projekt-Werkzeug normal geöffnet werden, werdem im Modus „Nicht vorübergehend“ geöffnet. Diese Dateien bleiben solange offen, bis sie ausdrücklich geschlossen werden. Vorübergehende Dateien, die bearbeitet wurden, werden automatisch in nicht vorübergehende Dateien umgewandelt.

Der Modus einer Datei kann zwischen „Vorübergehend“ und „Nicht vorübergehend“ gewechselt werden, indem das Stick-Pin-Symbol in der oberen rechten Ecke des Editor-Bereiches angeklickt wird. Klicken Sie mit der rechten Maustaste auf das Stick-Pin-Symbol, um zwischen den zuletzt besuchten Dateien zu navigieren (blaue Einträge sind vorübergehende Dateien, schwarze Einträge sind nicht-vorübergehend Dateien).

## 4.6. Strukturelles Falten

Der Editor unterstützt optional strukturelles Falten für Python, C, C++, Java, Javascript, HTML, Eiffel, Lisp, Ruby und eine Reihe anderer Dateiformate. Dies ermöglicht Ihnen, logische, hierarchische Abschnitte Ihres Codes visuell zusammenzuklappen, wenn Sie in anderen Teilen der Datei arbeiten.

Sie können Strukturelles Falten als Ganzes mit der Einstellung **Falten aktivieren** an-

und ausschalten. Wenn Falten angeschalten ist, können individuelle MIME-Typen mit der Einstellung **MIME-Typen falten** ausgeschalten werden (aber das Hinzufügen eines MIME-Typen hier, fügt nicht automatisch das Falten für diesen MIME-Typ hinzu).

Der **Zeilenmodus** in den Einstellungen für das Falten kann verwendet werden, um zu bestimmen, ob an einem Faltepunkt eine waagerechte Linie gezeichnet wird, ob diese über oder unter dem Faltepunkt gezeichnet wird und ob sie angezeigt wird, wenn der Faltepunkt zusammengeklappt oder erweitert ist. Der **Indikatorstil** wird genutzt, um das Aussehen der Faltmarkierungen, die an Faltepunkten angezeigt werden, auszuwählen.

Wenn das Falten angeschalten ist, erscheint auf der linken Seite der Source-Dateien ein zusätzlicher Rand, auf dem Faltepunkte angezeigt werden. Klicken Sie mit der linken Maustaste auf eine dieser Markierungen, um diesen Faltepunkt zusammenzuklappen oder zu erweitern.

Sie können auch die folgenden Tastenkombinationen gedrückt halten, während Sie klicken, um das Verhalten des Faltens zu ändern:

- **Umschalttaste** -- Wird die Umschalttaste beim Klicken auf einen Faltepunkt gedrückt gehalten, wird dies den Punkt und alle seine Kinder rekursiv erweitern, so dass die maximale Ebene der Erweiterung im eins erhöht wird.
- **Strg** -- Wird die Strg-Taste beim Klicken auf einen Faltepunkt gedrückt gehalten, wird dies den Punkt und alle seine Kinder rekursiv zusammenklappen, so dass die maximale Ebene der Erweiterung um eins verringert wird.
- **Strg+Umschalttaste** -- An einem gegenwärtig erweiterten Faltepunkt wird dies alle Kind-Haltepunkte rekursiv bis zur maximalen Tiefe zusammenklappen, genauso wie den äußeren. Wenn der Faltepunkt nachfolgend wieder mit einem normalen Klick erweitert wird, werden seine Kinder zusammengeklappt erscheinen. Strg-Umschalt-Klick auf einen zusammengeklappten Faltepunkt wird rekursiv die erneute Erweiterung aller Kinder bis zur maximalen Tiefe erzwingen.

Faltebefehle sind auch im Abschnitt Strukturelles Falten im Menü Source und über die angegebenen Tastaturkombinationen verfügbar:

- **Aktuelle Falte wechseln** -- Wie das Klicken auf den Faltrand, bearbeitet dies den ersten Faltepunkt, der in der aktuellen Auswahl oder auf der aktuellen Zeile gefunden wird.
- **Aktuelle mehr zusammenklappen** -- Wie Strg-Klick klappt dies den aktuellen Faltepunkt um ein weiteres Level zusammen.
- **Aktuelle mehr erweitern** -- Wie Umschalt-Klick erweitert dies den aktuellen Faltepunkt um ein weiteres Level.

- **Aktuelle vollständig zusammenklappen** -- Wie Umschalt-Strg-Klick auf einen erweiterten Knoten, klappt dies alle Kinder rekursiv bis zur maximalen Tiefe zusammen.
- **Aktuelle vollständig erweitern** -- Wie Umschalt-Strg-Klick auf einen zusammengeklappten Knoten, stellt dies sicher, dass alle Kinder rekursiv bis zur maximalen Tiefe erweitert werden.
- **Alle Zusammenklappen** -- Klappt die gesamte Datei rekursiv ohne Bedingungen zusammen.
- **Alle Erweitern** -- Erweitert die gesamte Datei rekursiv ohne Bedingungen.

## 4.7. Klammernsuche

Wing wird zusammenpassende Klammern in grün markieren, wenn der Cursor neben einer Klammer ist. Nicht zusammenpassende Klammern werden in rot markiert.

Sie können Wing dazu veranlassen, die gesamten Inhalte des innersten Klammernpaares von der aktuellen Cursor-Position zu markieren, indem Sie den Eintrag Klammernsuche aus dem Menü Source auswählen.

Für runde Klammern, eckige Klammern und geschweifte Klammern wird in allen Dateien die dazugehörige Klammer gesucht. Bei spitzen Klammern (`<` und `>`) wird auch in HTML- und XML-Dateien die entsprechende Klammer gesucht.

## 4.8. Einrückung

Einrückungen sind in Python syntaktisch bedeutend. Daher stellt Wing viele Funktionen zum Prüfen und Verwalten von Einrückungen im Source-Code bereit.

**Einrückungseinstellungen**

Die folgenden Einstellungen bestimmen, wie sich Einrückungsfunktionen in *neu erstellten* Source-Dateien verhalten.

1) Die Einstellung **Tabgröße** definiert die Standardgröße für jedes Tabzeichen in Leerzeichen.
2) Die Einstellung **Einrückungsgröße** definiert die Standardgröße für jedes Einrückungslevel in Leerzeichen. Dies kann in Dateien, die nur Tabs in den

Einrückungen enthalten, geändert werden, um es ein Vielfaches der konfigurierten Tabgröße zu machen.

3) Die Einstellung **Einrückungsstil** definiert den Standard-Einrückungsstil, entweder `Nur-Leerzeichen`, `Nur-Tabs` oder `Gemischt`. Die gemischte Einrückung ersetzt alle Leerzeichen in Tabgröße mit einem Tabzeichen.

Diese Einstellungen definieren, wie Einrückungen vom Editor behandelt werden:

4) Die Einstellung **Automatisch Einrücken** kontrolliert, ob jede neue Zeile automatisch eingerückt wird.

5) Die Einstellung **Einrückungslinien anzeigen** steuert, ob Einrückungslinien als dünne senkrechte Linien angezeigt werden. Dieser Wert kann auf einer Pro-Dateibasis außer Kraft gesetzt werden, und zwar im Editor-Reiter unter Dateieigenschaften.

### Bestimmung des Éinrückungsstils

Wird eine bestehende Datei geöffnet, wird sie durchsucht, um die in dieser Datei verwendete Einrückungsart zu bestimmen. Wenn die Datei Einrückungen enthält, kann dies die Werte für Tabgröße, Einrückungsgröße und den Einrückungsstil, die in den Einstellungen festgelegt sind, außer Kraft setzen und die Einrückungen in der Datei werden so vorgenommen, dass sie mit dem bestehenden Inhalt anstatt mit den konfigurierten Standardwerten übereinstimmen. Wenn gemischte Formen der Einrückung gefunden werden, wird die allgemeinste Form verwendet.

In Python-Dateien kann die ermittelte Einrückungsform nicht außer Kraft gesetzt werden und Wing wird neue Einrückungen immer an die bestehenden Einrückungen anpassen. Gemischte Einrückungsstile in Python sind gefährlich, da Einrückungen syntaktische Bedeutung haben.

In nicht-Python-Dateien können Sie den Einrückungsstil schnell ändern, indem Sie im Dialog `Dateieigenschaften` die Funktion `Einrückungsstil` auswählen. Dies ermöglicht das Erstellen von Dateien, die absichtlich Einrückungsformen in verschiedenen Teilen der Datei mischen. Wenn Sie möchten, dass Wing zu der Einrückungsform zurückkehrt, die es in der Datei als am bedeutendsten bestimmt, wählen Sie den Eintrag `Mit der Datei übereinstimmenden Stil verwenden`.

Sie können auch die gesamte Datei in verschiedene Einrückungsformen umwandeln, indem Sie im Menü Source im Abschnitt Einrückungen den Einrückungsmanager verwenden. Dies wird im Abschnitt **Einrückungsmanager** beschrieben.

**Tabgröße**
Die Tabgröße wird für alle Python-Source-Dateien, die Leerzeichen in der Einrückung enthalten, automatisch auf 8 Zeichen gezwungen. Dies wird gemacht, da der Python-Interpreter Tabs als 8 Zeichen definiert, wenn diese zusammen mit Leerzeichen verwendet werden. Diese Version von Wing erkennt Tabgrößenkommentare im `vi`-Stil nicht, aber es wendet die Einstellung **Tabgröße** an, wenn eine Datei nur Tabs in den Einrückungen enthält oder wenn es eine nicht-Python-Datei ist.

**Einrückungsgrundsätze**

Der Projektmanager erlaubt, den bevorzugten Einrückungsstil zu definieren (überschreibt den in Einstellungen definierten Stil) und ermöglicht außerdem, einen Grundsatz für das Erzwingen von Zeilenenden auf einer Pro-Projektbasis zu bestimmen. Dies wird mit den Einstellungen **Bevorzugtes Zeilenende** und **Zeilenenden-Grundsatz** unter Optionen in den Projekteigenschaften erreicht.

### 4.8.1. Automatisch Einrücken

Bei der Lieferung des IDE's ist die Funktion Automatisch einrücken angeschalten. Dies verursacht, dass Leerräume am Anfang von jeder neu erstellten Zeile hinzugefügt werden, wenn die Return-Taste oder die Eingabetaste betätigt wird. Es wird genug Leeraum eingefügt, um die Einrückung an das Einrückungslevel der vorherigen Zeile anzupassen und möglicherweise wird ein Einrückungslevel hinzugefügt oder entfernt, wenn dies durch den Kontext des Source-Codes deutlich wird (zum Beispiel `if`, `while` oder `return`).

Beachten Sie, dass wenn die Einstellung **Automatisch einrücken** auf `Falsch` eingestellt ist, erst bei Betätigung der Tab-Taste automatisch eingerückt wird.

### 4.8.2. Die Tab-Taste

Standardmäßig verhält sich die Tab-Taste genauso wie das automatische Einrücken: Der Leerraum am Anfang der aktuellen Zeile wird angepasst, um ein vernünftiges Einrückungslevel für diese Zeile zu erreichen.

Bestehender Leerraum am Anfang wird durch einen Leerraum ersetzt, der entweder nur Leerzeichen oder Tabs und Leerzeichen enthält, wie durch die oben beschriebene Methode bestimmt. Dieses Verhalten kann auch das Einrückungslevel einer Zeile verringern, wenn es entsprechend seinem Kontext als zu weit eingerückt erachtet wird.

Wenn beim Drücken der Tab-Taste mehrere Zeilen markiert sind, werden alle diese Zeilen als eine Einheit ein- oder ausgerückt, entsprechend der Änderung, die für die erste Zeile

der gewählten Einheit notwendig ist. Dies ist sehr hilfreich, wenn Böcke von Code verschoben werden.

Um ein echtes Tabzeichen einzufügen, ungeachtet des Einrückungsmouds oder der Position des Cursors in einer Zeile, tippen Sie Strg-Tab oder Strg-T.

### 4.8.3. Einrückung überprüfen

Wing IDE analysiert bestehende Einrückungen immer wenn eine Python-Source-Datei geöffnet wird und zeigt problematische Mischungen von Einrückungsstilen an. Dies ermöglicht Ihnen, die Datei zu reparieren. Dateien können jederzeit mit dem **Einrückungsmanager** näher geprüft oder repariert werden.

Wing zeigt auch verdächtige, nicht übereinstimmende Einrückungen im Source-Code an, indem es den Einrückungsbereich der relevanten Zeilen in blau unterstreicht.

> Im Allgemeinen kann es verwirrend sein, Tab/Leerzeichen- und Nur-Leerzeichen-Einrückungen in der gleichen Datei zu mischen, insbesondere wenn Dateien mit unterschiedlichen Editoren und von unterschiedlichen Entwicklern angesehen werden. Es wird daher empfohlen, entweder nur Leerzeichen oder nur Tabs zu verwenden. Verwenden Sie den Einrückungsmanager, um bestehenden Code, der eine Mischung aus Tabs und Leerzeichen enthält, umzuwandeln.

### 4.8.4. Blockeinrückung ändern

Wing stellt im Einrückungsteil des Menüs Source Befehle zum Einrücken und Ausrücken bereit, um das Erhöhen oder Verringern des Einrückungslevels von markierten Textblöcken zu unterstützen. Alle Zeilen, die in die aktuelle Textauswahl einbezogen sind, werden verschoben, selbst wenn nicht die gesamte Zeile markiert ist.

Einrückungen, die durch diese Befehle gesetzt werden, enthalten entweder nur Leerzeichen, nur Tabs oder eine Mischung aus Tabs und Leerzeichen, wie durch die im Kapitel **Einrückung** beschriebene Methode bestimmt.

### 4.8.5. Einrückungsmanager

Der Einrückungsmanager (im Menü `Werkzeuge`) kann verwendet werden, um Einrückungsstile in Source-Dateien zu prüfen und zu ändern. Er besteht aus zwei Teilen: (1) Dem Einrückungsbericht und (2) dem Einrückungskonvertierer.

Ein Bericht über die Art von bestehenden Einrückungen, die in Ihrer Source-Datei gefunden werden, ist über der horizontalen Teilungslinie gegeben. Er beinhaltet die Anzahl der gefundenen Nur-Leerzeichen, Nur-Tabs und Gemischte Tabs- und Leerzeichen-Einrückungen, Informationen darüber, ob die Einrückung in der Datei Probleme mit dem Python-Interpreter verursachen kann, und die Tab- und Einrückungsgröße, die für diese Datei berechnet wurde. Der Manager stellt auch Informationen darüber bereit, woher die berechneten Werte für die Tab- und Einrückungsgröße kommen (zum Beispiel führt eine leere Datei zur Verwendung der Voreinstellungen, die in den Einstellungen konfiguriert sind).

Umwandlungsoptionen für Ihre Datei sind unter der horizontalen Teilungslinie zu finden. Die drei Reiter werden verwendet, um den gewünschten Umwandlungstyp zu bestimmen. Jeder Reiter enthält Informationen über die Verfügbarkeit und Aktion dieser Umwandlung und eine Schaltfläche zum Starten der Umwandlung. Die meisten dieser Umwandlungen haben keine Parameter, die vom Nutzer geändert werden können. Nur für die Umwandlung vom Einrückungsstil Nur-Tabs in Nur-Leerzeichen kann der Wert für die Tabgröße, der im Einrückungsbericht angezeigt ist, bearbeitet werden. Dies setzt den konfigurierten Standardwert außer Kraft.

Sobald die Umwandlung abgeschlossen ist, wird der Einrückungsmanager aktualisiert, um den neuen Status der Datei und Aktionen von nachfolgenden Umwandlungen anzuzeigen.

## 4.9. Auto-Vervollständigung

Während Sie Python-Source-Code eingeben, wird Wing ein Popup für die Auto-Vervollständigung anzeigen, das verwendet werden kann, um die Tipparbeit zu reduzieren. Um davon Gebrauch zu machen, tippen Sie solange, bis das korrekte Symbol in der Liste markiert ist und drücken dann die Tab-Taste. Wing wird die verbleibenden Zeichen für das Source-Symbol ergänzen und eventuelle Rechtschreibfehler, die Sie in dem Namen gemacht haben, korrigieren.

Wenn Sie einen Namen auswählen möchten, ohne genug Zeichen einzugeben, die diese Auswahl für den Auto-Vervollständiger eindeutig machen, können Sie auch die Pfeiltasten nach oben und unten auf der Tastatur oder die Maus verwenden, um in der Popup-Liste nach oben oder unten zu rollen. Drücken Sie die Tab-Taste oder doppelklicken Sie auf den Listeneintrag, um das Symbol in Ihrem Source-Code zu vervollständigen.

Um das Popup des Auto-Vervollständigers zu verlassen, klicken Sie auf die `Esc`-Taste oder verwenden Sie `Strg-g`. Der Auto-Vervollständiger wird auch verschwinden, wenn Sie das Source-Symbol verlassen (zum Beispiel indem Sie ein `Leerzeichen` oder irgendein anderes Zeichen, das nicht in einem Source-Symbol enthalten sein kann, drücken) oder wenn Sie andere tastaturgebundene Befehle erteilen, die vom Auto-Vervollständiger

nicht akzeptiert werden (zum Beispiel `Speichern` durch die Tastenkombination oder rechte/linke Pfeiltaste).

> **Beschränkungen des Auto-Vervollständigers**
> Die Auto-Vervollständigung deckt momentan die meisten, aber nicht alle möglichen Szenarios ab. Lesen Sie den Abschnitt **Source-Code-Analyse** für zusätzliche Informationen über die gegenwärtigen Fähigkeiten.

## 4.10. Source-Assistent

Wing's `Source-Assistent` Werkzeug kann verwendet werden, während Source-Code angezeigt oder bearbeitet wird, um zusätzliche Informationen über den Definitionspunkt von Source-Konstrukten, die in der Nähe der aktuellen Einfügungscursor-Position liegen, zu sehen.

Die folgenden Daten werden im Source-Assistenten angezeigt:

- **Datei** -- Die Datei und Zeilennummer, wo das Source-Symbol definiert ist.
- **Call-Signatur** -- Der Funktions- oder Methodenname, Argumente und der Return-Wert, wenn bekannt, für das Source-Konstrukt. Der Klassenname ist auch enthalten, wenn verfügbar. Dieser Wert ist bei non-callable Werten leer.
- **Doc-String** -- Der Dokumentationsstring für das Source-Symbol, wenn verfügbar.

Sehen Sie sich das folgende Beispiel an, bei dem `|` am Ende von `myfile2` den Einfügungscursor darstellt:

**myfile1.py**:

```
class A:
  """Dies ist eine Beispiel-Klasse"""

  def meth(self, x):
    """Dies ist eine Beispiel-
Methode, die einen Parameter außer self
    akzeptiert und ein Dictionary ausgibt"""

    return {}
```

**myfile2.py**:

```
import myfile1
a = myfile1.A()
a.meth|
```

Nachdem der Nutzer `a.meth` eingegeben hat, wird Wing in diesem Fall den Doc-String und die Call-Signatur für die `meth` Methode in Klasse `A` heraussuchen und Informationen darüber im Source-Assistenten anzeigen:

```
Datei:              myfile1.py, line 4
Call-Signatur:      A.meth(self, x) -> dict
Doc-String:         Dies ist eine Beispiel-
Methode, die einen Parameter außer
                    self akzeptiert und ein Dictionary ausgibt
```

Beachten Sie, dass Wing nicht die Arten aller Argumente oder Return-Werte bestimmen kann, aber es präsentiert so viele Informationen wie es vom Source-Code herausfinden kann. Um Wing bei der Erstellung einer vollständigeren Analyse Ihres Source-Codes zu unterstützten, können Sie Statements wie die folgenden hinzufügen, um Hinweise auf die Arten der Werte bereitzustellen:

```
assert isinstance(myvalue, mymodule.CMyClass)
```

Für Erweiterungsmodule, die in C/C++ geschrieben sind, kann Wing eine Interface-Datei gegeben werden, die ein Python-Skeleton ist, welches die vom Erweiterungsmodul definierten Funktionen, Attribute, Klassen und Methoden wiederholt. Diesen Dateien sollte der Name des Erweiterungsmoduls plus `*.pi` gegeben werden. Zum Beispiel würde die Interface-Datei für ein Erweiterungsmodul, welches als `mymodule` importiert wird, als `mymodule.pi` bezeichnet werden. Beispiele von Interface-Dateien können in `resources/builtin-pi-files` innerhalb Ihrer Wing IDE Installation gefunden werden.

## 4.11. Automatisch speichern

Der Source-Code-Editor speichert Dateien alle paar Sekunden automatisch auf dem Laufwerk. Die automatisch gespeicherten Dateien werden in einem Unterverzeichnis Ihres **Verzeichnisses der Benutzereinstellungen** platziert.

Wenn Wing jemals abstürzt oder von außerhalb abgebrochen wird, können Sie diese Dateien verwenden, um alle ungespeicherten Änderungen wiederherzustellen. Kopieren Sie diese automatisch gespeicherten Dateien, um die älteren ungespeicherten Dateien zu überschreiben, aber führen Sie zuerst einen Vergleich durch, um sicherzustellen, dass die automatisch gespeicherten Dateien die sind, die Sie möchten.

## 4.12. Hinweise zu Kopieren/Einfügen

Es gibt viele Wege, um Text im Editor zu kopieren und einzufügen:

- Verwenden Sie die Einträge des Menüs Bearbeiten. Dies speichert den Text von Kopieren/Ausschneiden in der systemweiten Zwischenablage und kann in andere Anwendungen eingefügt oder von anderen Anwendungen kopiert werden.
- Verwenden Sie die im Menü Bearbeiten definierten Tastenkombinationen.
- Klicken Sie mit der rechten Maustaste auf die Oberfläche des Editors und verwenden die Einträge aus dem Popup-Menü, das erscheint.
- Wählen Sie einen Textbereich und ziehen ihn mit der Funktion 'Ziehen und Ablegen' (Drag and Drop) (das Drücken der Umschalttaste vor dem Ablegen verschiebt den Text anstatt ihn zu kopieren).
- In Linux: Markieren Sie Text irgendwo auf dem Bildschirm und klicken dann mit der mittleren Maustaste, um ihn am Punkt des Klicks einzufügen.
- Im Emacs-Modus: Die Tastenkombination Strg-k (`kill-line`) wird jeweils eine Zeile in die private Emacs-Zwischenablage ausschneiden. Dies wird separat von der systemweiten Zwischenablage gehalten und wird mit der Tastenkombination Strg-y (`yank-line`) eingefügt. In Windows und Mac OS X wird Strg-y die Inhalte der systemweiten Zwischenablage nur dann einfügen, wenn die Emacs-Zwischenablage leer ist.
- In Windows und Mac OS X: Klicken Sie mit der mittleren Maustaste, um die aktuelle, private Emacs-Zwischenablage (wenn in Emacs-Modus und die Ablage ist nicht-leer) oder die Inhalte der systemweiten Zwischenablage (in allen anderen Fällen) einzufügen. In Mac OS X wird die mittlere Maustaste nachgebildet, indem Sie während des Klickens die Programmsteuertaste gedrückt halten.

Es ist wichtig zu beachten, welche Aktionen die systemweite Zwischenablage verwenden, welche die Emacs-Zwischenablage nutzen (nur Emacs-Modus) und welche die X Windows-Auswahl verwenden (nur X Windows). Ansonsten sind diese Befehle in ihren Wirkungen austauschbar.

## 4.13. Geänderte Dateien automatisch Neuladen

Wing's Editor erkennt, wenn Dateien außerhalb des IDEs geändert wurden und kann Dateien automatisch oder nach der Aufforderung für Ihre Erlaubnis neu laden. Dies ist

hilfreich, wenn Sie mit einem externen Editor arbeiten oder wenn Sie Werkzeuge zur Erzeugung von Code verwenden, die Dateien neu schreiben.

Das Standardverhalten von Wing ist es, extern geänderte Dateien, die Sie noch nicht innerhalb von Wing's Source-Editor geändert haben, automatisch neu zu laden und bei Dateien, die auch innerhalb des IDEs geändert wurden, zum Neuladen aufzufordern.

Sie können dieses Verhalten ändern, indem Sie die Werte der Einstellungen **Neuladen wenn Unverändert** und **Neuladen wenn Geändert** einstellen.

In Windows verwendet Wing ein Signal vom Betriebssystem, um Änderungen zu erkennen, so dass die Benachrichtigung oder das Neuladen normalerweise sofort erfolgen. In Linux und Unix fragt Wing das Laufwerk standardmäßig alle 3 Sekunken ab; diese Frequenz kann mit der Einstellung **Externe Prüffrequenz** geändert werden.

## 4.14. Suchen/Ersetzen

Wing stellt eine Vielzahl von Werkzeugen für das Suchen und Ersetzen in Ihrem Source-Code bereit. Welche Sie verwenden, hängt von der Komplexität Ihrer Suchen- oder Ersetzen-Aufgabe ab und davon, mit welchem Stil des Suchens Sie am vertrautesten sind.

### 4.14.1. Schnellsuche mit der Werkzeugleiste

Eine Möglichkeit, einfache Suchen durchzuführen, besteht darin, Text in das Suchfeld der Werkzeugleiste einzugeben. Während Sie Text eingeben, wird zum nächsten Treffer, der nach der aktuellen Cursor-Position gefunden wird, gerollt. Das Drücken von `Enter`, sucht jeweils nach dem folgenden Treffer und setzt die Suche am Anfang des Dokumentes fort, wenn das Ende der Datei erreicht ist.

Die Textübereinstimmung bei der Schnellsuche mit der Werkzeugleiste ist von der Groß- und Kleinschreibung unabhängig, es sei denn, Sie geben einen Großbuchstaben als Teil Ihrer Suchzeichenkette ein.

Wenn der Fokus nicht auf dem Suchfeld der Werkzeugleiste liegt und es bereits eine Suchzeichenkette enthält, dann wird, wenn darauf geklickt wird, die Suche nach dem nächsten Treffer sofort im aktuellen Source-Editor starten. Wenn Sie stattdessen nach einer anderen Zeichenkette suchen möchten, dann löschen Sie den Text und geben die gewünschte Suchzeichenkette ein. Während Sie löschen, wird sich die Trefferposition im Editor rückwärts bewegen, bis sie die ursprüngliche Startposition Ihrer Suche erreicht, so dass Ihnen nach dem Eingeben Ihrer neuen Suchzeichenkette der erste Treffer nach der ursprünglichen Cursor-Position des Source-Editors angezeigt wird.

### 4.14.2. Tastaturgesteurte Mini-Suche/Ersetzen

Das Bearbeiten-Menü enthält ein Untermenü mit dem Namen Mini-Suche, welches die verfügbaren tastaturgesteuerten Suchoptionen aufzählt. Diese werden normalerweise mit den Tastaturbefehlsfolgen, die im Menü angezeigt werden, ausgelöst und können vollständig mit der Tastatur gesteuert werden. Die gesamte Interaktion mit dem Mini-Suchmanager erfolgt unter Verwendung eines Dateneingabebereiches, der nach Bedarf im unteren Teil des IDE-Fensters angezeigt wird.

Die Implementation des Mini-Suchmanagers ist den allgemein verwendeten Suchen- und Ersetzen-Funktionen, die in Emacs gefunden werden, sehr ähnlich, aber sie ist immer verfügbar, egal ob die Emacs Editor-Individualität verwendet wird oder nicht.

Die folgenden Suchen- und Ersetzen-Funktionen stehen für diese Einrichtung zur Verfügung:

- **Vorwärts** und **Rückwärts** -- Diese Optionen zeigen im unteren Teil des IDE-Fensters ein Eingabefeld für die Suchzeichenkette an und suchen in dem aktuellen Source-Editor interaktiv vorwärts oder rückwärts, wobei die Suche an der aktuellen Cursor-Position beginnt. Die Suche findet statt, während Sie tippen und kann mit `Esc` oder `Strg-G` abgebrochen werden, wodurch der Editor zu seinem ursprünglichen Cursor-Standort und seiner ursprünglichen Rollposition zurückkehrt.

  Die Suche ist von der Groß-/Kleinschreibung unabhängig, es sei denn, Sie geben einen Großbuchstaben als Teil Ihrer Suchzeichenkette ein. Um wiederholt zu suchen, drücken Sie `Strg-U` (`Strg-S` in Emacs-Tastaturmodus), um vorwärts zu suchen, und `Strg-Umschalt-U` (`Strg-R` in Emacs-Modus), um rückwärts zu suchen. Die Suchrichtung kann beliebig oft geändert werden und die Suche wird immer dann umbrechen, wenn der Anfang oder das Ende der Datei erreicht ist. Sie können `Strg-U` (`Strg-S` in Emacs-Modus) oder `Strg-Umschalt-U` (`Strg-R` in Emacs-Modus) auch am Anfang eingeben, wenn das Eingabefeld für den Suchbefehl noch leer ist, um die zuletzt verwendete Suchzeichenkette aufzurufen und die Suche (vorwärts oder rückwärts) mit dieser zu beginnen.

- **Auswahl Vorwärts** und **Auswahl Rückwärts** -- Diese Optionen funktionieren wie die obigen, aber sie beginnen mit der Auswahl im aktuellen Source-Editor.

- **Abfragen/Ersetzen** -- Diese Option verlangt eine Suchen- und eine Ersetzen-Zeichenkette in einem Eingabefeld am unteren Ende des IDE-Fensters und fragt bei jedem einzelnen Treffer, der nach der Cursor-Position im aktuellen Source-Editor gefunden wird, ob dieser ersetzt werden soll. Drücken Sie `y` zum Ersetzen und `n`, um einen Treffer zu überspringen und zum nächsten weiterzugehen. Die Interaktion kann jederzeit mit `Esc` oder `-G` abgebrochen werden. Die Übereinstimmungen sind von Groß-/Kleinschreibung unabhängig, es sei denn, ein Großbuchstabe wird als Teil der Suchzeichenkette eingegeben. Die Suche erfolgt immer vorwärts und

stoppt, wenn das Ende der Datei erreicht wird; es erfolgt kein Umbruch zu undurchsuchten Teilen, die zwischen dem Anfang der Datei und der Position, an der die Suche gestartet wurde, liegen.

- **Zeichenkette ersetzen** -- Dies funktioniert wie der obige Befehl, aber ersetzt alle Treffer sofort, ohne nachzufragen.

### 4.14.3. Suchen/Ersetzen-Werkzeug

Das andockbare Suchen/Ersetzen-Werkzeug kann für fortgeschrittene Suchen- und Ersetzen-Aufgaben verwendet werden. Sie können bestimmen, ob Groß- und Kleinschreibung beachtet werden soll und ob eine vollständige oder teilweise Wortübereinstimmung erfolgen soll. Das Werkzeug erlaubt außerdem das Suchen unter Verwendung von Wildcard- oder Regex-Suchzeichenketten (mit der entsprechenden Regex-Ersetzen-Fähigkeit) und unterstützt die Batch-Suche aller geöffneten Dateien, aller Projektdateien oder Datei-Sets auf dem Laufwerk.

#### 4.14.3.1. Modi und Bereich für Suchen/Ersetzen

Bevor Sie eine Suche beginnen, müssen Sie den Suchmodus und -bereich Ihrer Suche aus den Popup-Menüs, die sich oben im Fenster des Suchen/Ersetzen-Werkzeuges befinden, auswählen. Die folgenden Suchmodi stehen Ihnen zur Verfügung:

- **Interaktive Suche** -- Dies durchsucht eine einzelne Source-Editor-Datei und beginnt dabei an der aktuellen Cursor-Position. Die Suchergebnisse werden einzeln angezeigt, indem Sie sie im Soure-Editor auswählen. Standardmäßig wird der aktuelle Editor durchsucht, aber Sie können das Popup-Menü `Bereich` verwenden, um aus anderen, derzeit geöffneten Dateien auszuwählen.

- **Batch-Suchdateien** -- Dieser Modus ermöglicht Ihnen, die Suchergebnisse in Listenform anzuzeigen, anstatt die Editoren jeweils nur nach einem Treffer zu durchlaufen. Der Suchbereich ist auf den aktuellen Editor voreingestellt, aber er kann auf jeden anderen offenen Editor geändert werden oder Sie können bestimmen, dass Alle geöffneten Dateien, Alle Projektdateien oder alle Projektdateien innerhalb eines vordefinierten Datei-Sets eingeschlossen werden.

  Wenn Sie im Batch-Modus durchsuchen, können Sie die Treffer der Suche im Source-Editor ansehen, indem Sie auf die Einträge in der Ergebnisanzeige klicken oder die Schaltflächen Vorwärts/Rückwärts verwenden. Beachten Sie, dass Dateien, die aufgrund einer solchen Suche geöffnet werden, vorübergehend sind und automatisch geschlossen werden, es sei denn, sie werden bearbeitet oder das Stick-Pin Symbol in der oberen rechten Ecke des Editor-Bereiches wird verwendet, um

diese Dateien solange als permanent zu markieren, bis sie vom Benutzer ausdrücklich geschlossen werden.

- **Batch-Suchlaufwerk** -- Dieser Modus ist dem obigen ähnlich, außer dass er auf Dateien innerhalb eines ausgewählten Verzeichnisses auf dem Laufwerk angewendet werden kann. Optional kann er rekursiv verwendet werden, um alle Dateien in einem Unterverzeichnis einzuschließen. Standardmäßig werden alle Dateien in die Suche eingeschlossen, aber dies kann auf eines der vordefinierten Datei-Sets begrenzt werden, indem Sie das Popup-Menü `Bereich` rechts neben dem Verzeichniseingabefeld verwenden.

Die Datei-Sets, die im Suchmanager verwendet werden, sind benutzerdefinierte Kriterien für die Einbeziehung oder den Ausschluss von Sets, die auf den Dateinamen angewendet werden, um zu bestimmen, ob eine Datei in die Suche eingeschlossen werden soll oder nicht. Siehe **Datei-Sets** für zusätzliche Informationen über deren Definition und Verwendung.

Beachten Sie, dass die Größe der Textbereiche für das Suchen und Ersetzen erweitert werden kann, indem Sie den Eintrag `Eingabefeld vergrößern` aus dem Popup-Menü `Historie` jeweils auf der rechten Seite auswählen.

#### 4.14.3.2. Optionen für Suchen/Ersetzen

Die folgenden Suchoptionen stehen Ihnen in dem Popup-Menü `Optionen` zur Verfügung:

- **Groß- und Kleinschreibung** -- Wählen Sie diese Option, um nur exakte Treffer von Groß- und Kleinbuchstaben in der Suchzeichenkette anzuzeigen.
- **Ganze Wörter** -- Markieren Sie diese Option, um zu bestimmen, dass Treffer von Leerräumen umgeben sind (Leerzeichen, Tabs oder Zeilenende).
- **Textsuche** -- Wählen Sie dies aus, um eine normale Textsuche ohne Wildcard oder Regex durchzuführen.
- **Wildcard-Suche** -- Wählen Sie dies, um die Verwendung von Sonderzeichen für Platzhalter in der Suchzeichenkette zu ermöglichen:
  - `*` kann verwendet werden, um eine Übereinstimmung mit jeder beliebigen Zeichenfolge, abgesehen von Zeilenenden, zu finden. Zum Beispiel würde die Suchzeichenkette `my*value` innerhalb einer einzelnen Textzeile alles das als Treffer anzeigen, dass mit `my` beginnt und mit `value` endet. Beachten Sie allerdings, dass `*` immer die längstmögliche Zeichenkette als Treffer anzeigt; d.h. `myinstancevalue = myothervalue` wird beispielsweise nur als ein Treffer angezeigt, anstatt als zwei Übereinstimmungen. Um dies zu vermeiden, verwenden Sie stattdessen die Regex-Suche mit `.*?` anstatt `*`.

  - ? kann verwendet werden, um ein einzelnes Zeichen, abgesehen von Zeilenenden, zu suchen. Zum Beispiel würde `my???value` jede Zeichenkette, die mit `my` beginnt, danach drei Buchstaben hat und mit `value` endet, als Treffer anzeigen.
  - [ and ] kann verwendet werden, um Sets von übereinstimmenden Zeichen anzuzeigen. Zum Beispiel wird `[abcd]` entweder `a`, `b`, `c` oder `d` als Treffer anzeigen. Die Angabe von `[a-zA-Z]` zeigt jeden Buchstaben im Bereich von `a` bis `z` (einschließlich) sowohl in Klein- als auch Großbuchstaben als Treffer an. Beachten Sie, dass die Bestimmung von Groß- und Kleinbuchstaben in Bereichen von Zeichen ignoriert werden, außer wenn Sie die oben beschriebene Option `Groß- und Kleinschreibung` aktiviert haben.

- **Regex-Suche** -- Wählen Sie dies aus, um als Suchstil einen regulären Ausdruck (regular expression) zu verwenden. Dies ist eine leistungsfähigere Variante als die Wildcard-Suche, denn sie ermöglicht komplexere Angaben der Suchtreffer und Ersetzungswerte. Informationen zur Syntax, die für die Such- und Ersetzungszeichenketten erlaubt ist, finden Sie in Python's Dokumentation zu Regulärer Ausdruckssyntax.

Die folgenden Optionen sind nur für die Suche in einer einzelnen Datei unter Verwendung des Interaktiven Suchmodus relevant. Batch-Suchen durchsuchen immer die gesamte Datei von oben bis unten:

- **Suche umbrechen** -- Heben Sie die Auswahl dieser Option auf, um das Umbrechen zu vermeiden, wenn die Suche den Anfang oder das Ende der Datei erreicht.
- **Zurück/Rückwärts** -- Wählen Sie dies, um aufwärts anstatt abwärts zu suchen.
- **Inkremental** -- Markieren Sie diese Option, um die Suche sofort, während Sie tippen oder die Suchoptionen ändern, zu starten oder neu zu starten. Wenn die Option nicht ausgewählt ist, müssen Sie die Schaltflächen Vorwärts/Rückwärts verwenden, um die Suche zu starten.

Die folgenden Optionen stehen in dem Popup-Menü `Andere Optionen` zur Verfügung, wenn Sie im Batch-Suchmodus arbeiten:

- **Rekursive Verzeichnissuche** -- Wählen Sie dies aus, um rekursiv innerhalb aller Unterverzeichnisse des gewählten Suchverzeichnisses zu durchsuchen.
- **Binärdateien auslassen** -- Wählen Sie dies, um alle Dateien, die Binärdaten enthalten, auszulassen.
- **Suchen automatisch neu starten** -- Wählen Sie diese Option, um die Suche sofort neu zu starten, wenn sie unterbrochen wurde, weil ein Suchparameter oder das Set der zu durchsuchenden Dateien geändert wurde.

- **Ersten Treffer öffnen** -- Wählen Sie dies, um den ersten Treffer der Batch-Suche automatisch zu öffnen, sogar bevor auf die Ergebnisliste geklickt wird.
- **Zeilennummern anzeigen** -- Wählen Sie dies, um die Zeilennummern in dem Batch-Ergebnisbereich anzuzeigen.
- **Dateiname des Resultats** -- Dies wird verwendet, um das Format von dem Dateinamen des Resultats, der in dem Batch-Ergebnisbereich angezeigt wird, zu bestimmen.

Die folgenden Optionen stehen zur Verfügung, wenn Sie im Ersetzen-Modus arbeiten:

- **Suchen nach Ersetzen** -- Wählen Sie dies, um automatich nach dem nächsten Treffer zu suchen, wenn Sie im interaktiven Ersetzen-Modus arbeiten.
- **Ersetzen arbeitet auf dem Laufwerk** -- Wählen Sie dies, um Text in ungeöffneten Dateien direkt auf dem Laufwerk zu ersetzen. Siehe **Ersetzen von Suchergebnissen** für Einzelheiten zu dieser Option.

#### 4.14.3.3. Suchergebnisse ersetzen

Bei Suchen, die an geöffneten Dateien arbeiten, erfolgt das Ersetzen immer im offenen Datei-Editor und kann später rückgängig gemacht oder auf dem Laufwerk gespeichert werden, so wie es für alle anderen Bearbeitungsvorgänge möglich ist.

Wenn Text im Batch-Modus ersetzt wird, kann es passieren, dass einige der durchsuchten Dateien gegenwärtig nicht in einem Editor geöffnet sind. In diesem Fall wird Wing standardmäßig alle geänderten Dateien öffnen und Änderungen in neu erstellten Editoren, die geöffnet bleiben, bis der Benutzer sie ausdrücklich speichert und schließt, vornehmen. Dies ist die sicherste Methode, globale Ersetzungsoperationen, die mehrere Dateien umfassen, durchzuführen, da deutlich gezeigt wird, welche Dateien geändert wurden und die Möglichkeit besteht, die Änderungen rückgängig zu machen.

Ein alternativer Ansatz besteht darin, die Option `Ersetzen arbeitet auf dem Laufwerk` aus dem Popup-Menü Optionen auszuwählen. Dies bewirkt, dass Wing Dateien direkt auf dem Laufwerk ändert, wenn es gegenwärtig keinen geöffneten Editor gibt.

Da es schwierig sein kann, globale Ersetzungsoperationen richtig durchzuführen, empfehlen wir ausdrücklich, ein Revisionskontrollsystem oder häufige Backups und manuell vergleichende Datei-Revisionen zu verwenden, bevor Sie Dateien akzeptieren, die geändert wurden.

## 4.15. User-defined Bookmarks

Wing IDE Professional and higher support named user-defined bookmarks that can be set and accessed from the Source menu and the key bindings shown there. Bookmarks names are global to the project and refer to a particular position within a selected file:

- **For Python files,** bookmarks are defined relative to the enclosing scope (method, class, or function), so edits before the line where the bookmark is located will usually not cause the bookmark's relative position in source code to be changed. Only edits between the anchoring scope, such as start of method, and the bookmarked line will cause a bookmark's position to slip. Wing currently does not try to track bookmarks when this is the case, but they can easily be redefined if exact location is important.
- **For all other types of files,** bookmarks are defined simply by file name and line number. If the file is edited, the bookmark's position may appear to slip.

When navigating to a bookmark from the Source menu or key binding, Wing will present a dialog or entry area at bottom of the screen (depending on editor personality) into which the bookmark name can be typed. A list of possible completions will be displayed. Pressing tab will select the currently highlighted completion.

A list of defined bookmarks is available in the Bookmarks tool, which is available from the Tools menu. Right click on an entry for a context menu of operations for the selected bookmark or bookmarks. Multi-selection is possible by holding down the shift and/or control keys. Double clicking or middle mouse clicking will navigate to the selected bookmark.

When the Bookmarks tool has focus, keyboard navigation is possible with the arrow keys and by typing letters to move quickly to a particular bookmark. Enter can then be pressed to navigate to the selected bookmark.

In VI mode, the standard `m` and `\'` plus key bindings are supported, in addition to the operations in the Source menu, which allow for the definition of bookmarks with names longer than one character.

Emacs, Brief, and other key bindings also support bookmarks. However, bookmark functionality for VI, Emacs, and Brief key bindings is omitted in Wing IDE Personal.

## 4.16. Templating (Code Snippets)

Wing provides support for defining and using templates for commonly reused bits of code (sometimes called code snippets) and other text. Templates might be used for standard

file skeletons, comment formats, dividers, class definitions, function definitions, HTML tables, and much more. Although Wing comes with a few example templates, in most cases users will want to define their own templates, to match their coding conventions and preferences.

Wing's templating facility is implemented using the **scripting sub-system** and is controlled from the **Templating** tool panel. In most cases, key bindings are assigned to templates so that they templating tool does not have to be visible in order to use a template.

## Overview

Templates are located either in `scripts/templates` inside the Wing IDE installation or in templates in the **user settings directory**. When a template of the same name is found in both, the template in the user settings directory will be used in preference whenever the template is referred to by name (for example, when assigning key bindings or invoking it with the `template` or `template-file` commands).

Advanced users can add additional template directories by altering `gTemplateDirs` global in the `templating.py` script.

Each template is in a file with name in the form `name.ext` where `name` is the name of the template and `ext` is the file extension that should be used when using the template to create a new file.

## Syntax

Templates are text that contains markers where user-provided values should be inserted. These markers are similar to Python's `%(varname)s` string substitution syntax but instead of containing only a variable name, the body of the marker contains richer argument collection information in the following format, with vertical bars dividing each value:

```
%(varname|type|default)s
```

Type and default are optional but the vertical bar must be present if omitting type but including a default. To write a template that includes Python style string formats, escape each `%` by writing `%%` instead.

Each part is defined as follows:

- **varname** -- The name of the variable. When the value is collected from the user,

underscores will be replaced by spaces and the words capitalized. For example, „user_name“ will be rendered „User Name“.

Any number of the following special characters may be prefixed to the variable name to control how it is used:

*Exclamation point (!)* indicates that the value should be shown for data collection even if a default value can be found for it. Otherwise, it is hidden when a default is found.

*At sign (@)* indicates that the value should be wrapped if it exceeds the configured **text wrap line column**.

- **type** -- The type of data to collect. Currently this is one of:

  *string(length)* -- a string with given maximum length (uses default 80 chars if length is omitted)

  *filename* -- a file name

  *date* -- current date in locale's preferred format or in the `time.strftime()` format given in the environment variable `__DATE_FORMAT__`

  *datetime* -- current date+time in locale's preferred format or in the `time.strftime()` format given in the environment variable `__DATETIME_FORMAT__`

  If this field is omitted or empty, string is assumed.

- **default** -- The default value to use. This may be the actual value, or may contain environment variable references in the form `$(envname)` to attempt to read all or part of the value from the named environment variable.

  Environment variables can be specified either in the `Debug` tab of Wing's **Project Properties** or in the environment that exists before Wing is launched. Values in the Project Properties override any values set before starting Wing.

  When this field is omitted, or when no default environment value can be found, the user will be prompted to enter the value.

## Indentation and Line Endings

Templates should always use one tab for each level of indentation. Tabs will be replaced with the appropriate indentation type and size when the template is used in a new or existing file (either according to content of the target file or using the configured **indent style** and **indent size** for new files). Wing will force tab indentation in all newly created template files.

Similarly, line endings in templates will be replaced with the appropriate type to match the file to which the template is applied. However, there is no requirement for template files to contain any particular kind of line ending.

If the template starts with `|x|` then `x` is a specification of how the indents in the template should be converted. It can be one of:

- *An integer*: Re-indent as a block, like Wing's indent-region command, so the first line is at the given number of indent levels.
- *The character 'm'*: Re-indent as a block, like Wing's indent-to-match command, so the first line is at the expected indent level according to context in the source.
- *The character 'm' followed by '+' or '-' and an integer*: Re-indent as for `'m'` and then shift left or right by the given number of indents.

Any `|x|` at the start of a template file will be removed before the template is inserted into an editor.

## Cursor Placement

Templates can contain `|!|` to indicate where the cursor should be placed once the template has been inserted into a file. This mark will be removed before templates are inserted into an editor.

## Reloading

The templating script will reload templates whenever they change on disk and will print warnings about any that cannot be parsed into the `Scripts` channel of the **Messages** tool.

## Commands

Once the templating support script has loaded into Wing, the following commands will be available for invoking templates:

- **template** -- This will insert a template (selected by name) at the cursor in the current editor. If there is a non-empty selection on the editor, it will replace the selection. The user will be prompted for any arguments defined by the template, if they cannot be found in defaults.

- **template-file** -- This will create a new file of the type specified by the template file's extension and insert the selected template into it, prompting the user as needed for arguments.

### User Interface

When templates are executed, Wing will prompt for any missing arguments found in the template (those for which no defaults can be determined; see above description of template syntax). Argument collection is achieved with the same built-in automatic argument collection engine that Wing uses to obtain missing command arguments, usually at the bottom of the current editor window.

The templating script also registers a new tool type with Wing IDE. The tool is not shown by default but can be inserted into Wing IDE dock windows from the `Tools` menu and the `Insert Tool` sub-menu of the tool area context menu. The tool panel currently supports adding, editing, removing, and executing templates, and also assigning key bindings for pasting selected templates into the current editor.

## 4.17. Using Revision Control with Wing

Wing integrates the most common revision control operations for both CVS (the Concurrent Versions System) and Subversion (a more modern replacement for CVS). It can be used to:

- Add files or directories to the repository
- Update to obtain changes from the repository
- Commit edits to the repository
- View differences between local copies and the repository or between most recent and previous repository version
- Obtain revision log and status information
- Under Subversion, list, blame/praise, and resolved are also supported

To turn on revision control for a project, use the **Extensions** tab in `Project Properties` and select the desired revision control system. A new menu `CVS` or `SVN` will appear in the menu bar and operations will be added to the editor and project manager context menus.

This integration assumes that you have already set up CVS or Subversion to work on your system and already have a repository checked out on your disk. If not, refer to the notes below.

Please send us suggestions, comments, or requests using the **Feedback** feature in the `Help` menu or by emailing to support at wingware dot com

## Installing CVS

### On Windows:

1. Download from http://www.nongnu.org/cvs

2. Add installation location to `PATH` environment variable from the Advanced tab of the System control panel

### On Linux:

1. Install CVS from using the packages that came with your Linux/Unix distribution or download from http://www.nongnu.org/cvs and build from sources.

## Installing Subversion

### On Windows:

1. Download from http://subversion.tigris.org/

2. Add installation location to `PATH` environment variable from the Advanced tab of the System control panel

### On Linux:

1. Install Subversion from using the packages that came with your Linux/Unix distribution or download from http://subversion.tigris.org/ and build from sources.

## Using SSH Repositories

Both CVS and Subversion can use SSH as a secure and convenient way to access the revision control repository.

**To set up SSH on Windows:**

1. Install putty -- the combined installer is easiest

2. Add the location where putty is installed to your `PATH` environment variable from the Advanced tab of the System control panel.

3. Run `puttygen` and generate an SSH2 RSA key pair. Use a passphrase you will remember. Save both private and public keys to disk. Copy the contents of the key box (starting with „ssh-rsa“) to `rsa-public.key` on disk.

4. Copy the `rsa-public.key` file to your server and add it to the `.ssh/authorized_keys` file under your username. E.g., use `pscp rsa-public.key user@hostname:` and then log into hostname and `cat rsa-public.key >> .ssh/authorized_keys`.

5. Run putty and enter host name in `Host Name` and `Saved Sessions` boxes then press `Save`. Go to the `Connection` category and enter your user name on the server into the `Auto-login` username box. Go back to `Session` category and press `Save` again.

6. Run `pageant`, which adds an icon to your Windows tray. Right click and select `Add Key`. Navigate to the private key saved from `puttygen` and enter your passphrase when prompted.

7. Restart `putty`, click on the saved session, press Load, and then Open. This should open a connection to the server without prompting for any further information.

**To set up SSH on Linux/Unix:**

If you do not already have `openssh` and `cvs` installed, install them from packages that came with your Linux or Unix distribution.

1. If `ssh-add -l` complains that it cannot find the SSH agent, run `ssh-agent bash` (or your favorite shell). This can be skipped on most modern Linux distributions because they run the X window manager inside ssh-agent.

2. If you don't already have an ssh key in `.ssh`, issue the command `ssh-keygen -t rsa` to create a key pair in `.ssh/id_rsa` (the private key) and `.ssh/id_rsa.pub` (the public key). Enter a passphrase you will remember.

3. Copy the file `.ssh/id_rsa.pub` to your server and add it to the `.ssh/authorized_keys` file under your username. E.g., use `scp rsa-public.key user@hostname:` and then log into hostname and `cat rsa-public.key >> .ssh/authorized_keys`.

4. Back on your client (where you plan to run Wing), type `ssh-add` and enter your passphrase to get the SSH key loaded into `ssh-agent`.

5. Type `ssh user@hostname` and you should be able to log into your server without being asked for a password.

## Subversion with SSH

**First time configuration:**

1. Install and configure SSH as described above (this also loads authentication information into the cache for the current session)

**To check out a repository:**

1. Type `svn checkout svn+ssh://hostname/path/to/repository`

   If you're not sure what to check out try this first:

   `svn list svn+ssh://hostname/`

**Future sessions require:**

1. **On Windows**, double click on your private key file and enter your pass phrase

   **On Linux/Unix**, run `ssh-add` and enter your pass phrase

2. Run Wing with a project where the **Enable Revision Control** property is set in the Extensions tab of Project Properties and Subversion is selected as the revision control system.

## Subversion with http/https or file URLs

**To check out a repository:**

1. **For http or https**, type `svn checkout http://hostname/path/to/repository`

   If you're not sure what to check out try this first:

   `svn list http://hostname/`

   **For file: URLs**, type `svn checkout file:///path/to/repository`

   You will be prompted for your user name and password, which will be cached by Subversion for future sessions.

**Future sessions require:**

1. Run Wing with a project where the **Enable Revision Control** property is set in the Extensions tab of Project Properties and Subversion is selected as the revision control system.

## Subversion without Authentication Cache

If you are using Subversion with http, https, or file access method and authentication cache disabled, you will need to go into the `Options` in the `SVN` menu in Wing (after enabling Subversion for your project) and select `Manual (from Wing)` for the `Authentication` option. This will prompt you for a user name and password the first time each repository is used in a Wing IDE session and sends them to Subversion with the `--username` and `--password` command line arguments, along with `no-auth-cache` so that your entries are never cached on disk.

## Using CVS with SSH

**First time configuration:**

1. Install and configure SSH as described above (this also loads authentication information into the cache for the current session)

2. **On Windows**, add `CVS_RSH=plink` to your environment from the Advanced tab of the System control panel.

   **On Linux/Unix**, add `CVS_RSH=ssh` to your environment. For example, `CVS_RSH=ssh; export CVS_RSH` on the command line, or add this to your `.bashrc` file.

Note that **Environment** in your Project Properties can also be used to set `CVS_RSH` or other environment variables, however only for CVS commands issued from the IDE.

**To check out a repository:**

1. Type `cvs -d :ext:username@hostname:/path/to/repository co module_name`

**Future sessions require:**

1. **On Windows**, double click on your private key file and enter your pass phrase

   **On Linux/Unix**, run `ssh-add` and enter your pass phrase

2. Run Wing with a project where the **Enable Revision Control** property is set in the Extensions tab of Project Properties and CVS is selected as the revision control system.

## Using CVS with pserver

CVS's pserver authentication mechanism is obsolete but it is still used for anonymous CVS access in some places, such as on sourceforge.net. If you are working with a pserver repository that requires a password (Sourceforge does not), then you will need to issue `cvs login` once from the command line before starting Wing.

## Notes on the Implementation

Wing's CVS and Subversion integration is based on the IDE's scripting extension API. Additional revision control systems can be added by basing on the `cvs.py` and `svn.py` sources found in the `scripts` directory within the Wing IDE installation.

If you plan to work on scripts that are in the `scripts` directory, copy them first to your **User Settings Directory** in the `scripts` directory there. When duplicate script names are found Wing will prefer those found in your user settings directory, so this allows you to make changes without losing those changes when Wing is updated in the future.

For more information on scripting, see **Scripting and Extending Wing IDE**.

# 4.18. Tastaturmakros

Das Menü Bearbeiten enthält Einträge zum Starten und Beenden der Definition eines Tastatur- oder Befehlssequenzmakros und Einträge für die Ausführung des zuletzt definierten Makros. Wenn die Makroaufzeichnung einmal gestartet ist, wird jeder Tastenanschlag oder Editor-Befehl als Teil dieses Makros aufgezeichnet, bis die Makroaufzeichnung wieder gestoppt wird. Die meisten Befehle sowie alle Zeicheneinfügungen und -löschungen können in Makros einbezogen werden.

**Wiederholte Makroausführung**
Im Emacs-Modus können Makros immer wieder ausgeführt werden, indem Sie escape eintippen, gefolgt von der Anzahl der Wiederholungen und gefolgt von der Tastenfolge der Makro-Ausführung. Geben Sie zum Beispiel `escape 1 0 strg-x e` ein, um ein Makro zehn Mal hintereinander auszuführen.

## 4.18.1. Beispiel eines Makros

Dieses Beispiel veranschaulicht die Verwendung von Tastaturmakros. Das gegebene Beispiel basiert auf der Verwendung des Emacs-Modus, da dies der Editor-Modus ist, der die meisten tastaturgesteuerten Befehle enthält, die sich für eine Kombination in nützlichen Makros eignen.

Eine häufige Aufgabe beim Schreiben von Python-Bindings für C/C++ Bibliotheken besteht darin, Listen von `#define` Konstanten zu kopieren und sie in Python-variable Zuweisungen umzuwandeln:

```
#define SC_MARK_CIRCLE 0
#define SC_MARK_ROUNDRECT 1
#define SC_MARK_ARROW 2
#define SC_MARK_SMALLRECT 3
#define SC_MARK_SHORTARROW 4
#define SC_MARK_EMPTY 5
#define SC_MARK_ARROWDOWN 6
#define SC_MARK_MINUS 7
#define SC_MARK_PLUS 8
```

Im Emacs-Modus kann das oben genannte umgewandelt werden, indem Sie den Cursor vor dem ersten `#define` positionieren, die Makrodefinition starten und die folgenden Tastenanschläge ausführen:

```
escape 8 strg-d strg-s <Leerzeichen> <rechte Pfeiltas-
te> = <Leerzeichen> strg-a <Pfeil nach unten>
```

Dies löscht die 8 Zeichen `#define` (mit nachfolgendem Leerzeichen) vor dem Cursor, springt zum Leerzeichen nach dem Konstanten-Identifier, fügt `=` ein und geht zum Anfang der nächsten Zeile. Wenn dies abgeschlossen ist, stoppen Sie die Makroaufzeichnung und tippen das Folgende ein, um die verbleibenden Zeilen umzuwandeln:

```
escape 8 strg-x e
```

Dies wird das Makro acht Mal ausführen und zu dem folgenden neu formatierten Source-Code führen (die erste Zeile wurde während der Erstellung des Makros neu formatiert):

```
SC_MARK_CIRCLE = 0
SC_MARK_ROUNDRECT = 1
SC_MARK_ARROW = 2
SC_MARK_SMALLRECT = 3
SC_MARK_SHORTARROW = 4
SC_MARK_EMPTY = 5
SC_MARK_ARROWDOWN = 6
SC_MARK_MINUS = 7
SC_MARK_PLUS = 8
```

Kombinieren Sie diese Technik mit Strg-Leerzeichen (set-mark-command) und führen Sie Kopieren/Einfügen aus, um die Reihenfolge der Konstrukte innerhalb einer Zeile zu ändern. Rückgängig kann verwendet werden, um Probleme, die durch fehlerhafte Makros oder eine falsche Cursor-Position vor der Ausführung des Makros verursacht wurden, zu beheben.

Makros werden beenden, wenn irgendein Befehl innerhalb des Makros fehlschlägt (zum Beispiel wenn eine zusätzliche Suche scheitert). Dies kann verwendet werden, um Bearbeitungen zu verhinden, wenn ein Makro an einer Stelle ausgeführt wird, die nicht sinnvoll ist.

## 4.19. Source-Code-Analyse

Wing's Auto-Vervollständiger, Source-Index-Menü, Gehe-zu-Definition Fähigkeiten, einige Funktionen zur Source-Neuformatierung sowie in Wing IDE Professional der Source-Code-Browser und Source-Assistent verlassen sich alle auf eine zentrale Maschine, die Ihren Source-Code im Hintergrund liest und analysiert, während Sie Dateien zu Ihrem Projekt hinzufügen oder Ihren Code im Source-Code-Editor ändern.

**So funktioniert die Analyse**
Für die Analyse Ihres Source-Codes wird Wing den Python-Interpreter und den `PYTHONPATH`, den Sie in Ihren **Projekteigenschaften** bestimmt haben, verwenden. Wenn Sie für Ihr Projekt eine Haupt-Debug-Datei festgelegt haben, dann werden die Eigenschaftswerte dieser Datei verwendet; andernfalls werden die projektweiten Werte verwendet. Wann immer sich irgendeiner dieser Werte ändert, wird Wing Ihren Source-Code vollständig neu analysieren.
Sie können den Python-Interpreter und `PYTHONPATH`, die von der Source-Code-Analyse-Maschine verwendet werden, ansehen, indem Sie den Eintrag `Analysestatistik anzeigen` aus dem Source-Menü auswählen. Die Werte in dem sich aufschlagenden Dialogfenster sind nur lesbar, aber wenn Sie auf die Schaltfläche Einstellungen klicken, können Sie Änderungen vornehmen. Siehe **Projektweite Eigenschaften** um Einzelheiten darüber zu erfahren, wie Sie diese Werte ändern.
Seien Sie sich bewusst, dass Wing bei der Verwendung von mehreren Versionen des Python-Interpreters oder unterschiedlicher `PYTHONPATH`-Werte für verschiedene Source-Dateien in Ihrem Projekt alle Dateien im Projekt analysieren wird und die Interpreter-Version und den `PYTHONPATH` verwenden wird, die es in der Haupt-Debug-Datei oder den projektweiten Debug-Eigenschaften findet. Dies kann zu fehlerhaften oder unvollständigen Analysen einiger Source-Dateien führen. Daher ist es das Beste, nur eine Python-Version mit jeder Wing IDE Projektdatei zu nutzen.

Die folgenden Punkte sind bekannte Beschränkungen, die Funktionen, welche auf der Source-Analyse basieren, beeinflussen:

- Die Analyse scheitert manchmal daran, den Typ eines Konstruktes zu identifizieren, weil der Python-Code nicht immer Anhaltspunkte zur Bestimmung des Datentyps bereitstellt. In diesen Fällen können Sie `isinstance` und/oder Interface-Dateien verwenden, um das Analyseprogramm zu informieren, wie weiter unter beschrieben.
- Typen von Elementen in Listen, Tuples und Dictionaries sind nicht identifiziert.
- Doc-Strings und andere Analyseinformationen können veraltet sein, wenn Sie eine Datei extern mit einem anderen Editor bearbeiten und diese in Wing nicht neu laden. Siehe Abschnitt **Geänderte Dateien automatisch Neuladen** für Optionen zum Neuladen.
- Einige neuere Python-Sprachenkonstrukte und mögliche Fälle der Typenanalyse werden nicht ausdrücklich unterstützt.

**Verwendung von isinstance() zur Unterstützung der Analyse**
Eine Möglichkeit, die Einrichtung der Code-Analyse über den Typ einer Variablen zu informieren, besteht darin, einen `isinstance`-Aufruf zu Ihrem Code hinzuzufügen. Ein Beispiel ist `assert isinstance(obj, CMyClass)`. Das Code-Analyseprogramm wird diese aufnehmen und vollständigere Informationen für diese Werte bereitstellen.

**Die Verwendung von *.pi Dateien zur Unterstützung der Analyse**
Wing's Source Analyser kann nur Python-Code lesen und umfasst keinen Support zum Verstehen für den Code von C/C++ Erweiterungsmodulen. Um die Code-Analyse über die Inhalte eines Erweiterungsmoduls zu informieren, ist es möglich, eine `*.pi` (Python-Interface) Datei zu erstellen. Zum Beispiel wird die Interface-Datei für ein Modul, das als `mymodule` importiert wird, als `mymodule.pi` bezeichnet. Diese Datei ist einfach ein Python-Skeleton mit der entsprechenden Struktur und Call-Signatur, damit sie mit den Funktionen, Attributen, Klassen und Methoden, die in einem Erweiterungsmodul definiert sind, übereinstimmt. In vielen Fällen können diese Dateien aus den Interface-Dateien automatisch erzeugt werden.
Wing sucht nach `*.pi` Dateien zuerst im gleichen Verzeichnis, in dem es das Erweiterungsmodul findet (oder im Verzeichnis des Source-Codes, wenn das Modul noch nicht kompiliert wurde und sich das Verzeichnis des Source-Codes in Ihrem konfigurierten Python Path befindet). Wenn die Dateien nicht gefunden werden, sucht Wing im Verzeichnispfad, für den die Einstellung **Schnittstellenpfad** gesetzt ist. Schließlich wird Wing im Verzeichnis `resources/builtin-pi-files` innerhalb Ihrer Wing IDE Installation suchen.
Bei der Suche im Schnittstellen-Pfad oder in Wing's Builtin-Verzeichnis wird zuerst die höchste Ebene des Verzeichnisses nach einer übereinstimmenden `*.pi` Datei durchsucht. Danach sucht Wing in einem Unterverzeichnis `#.#`, das entsprechend der Haupt- und Unterversion von Python, das mit Ihrer Source-Basis verwendet wird, benannt ist. Im Folgenden wird dann jede niedrigere Haupt-/Unterversion rückwärts bis `1.5` durchsucht.
Wenn sich zum Beispiel `c:\share\pi\pi-files` im Schnittstellenpfad befindet und Python 2.3 verwendet wird, sucht Wing zuerst in `c:\share\pi\pi-files`, dann in `c:\share\pi\pi-files\2.3`, danach in `c:\share\pi\pi-files\2.2` und so weiter.
Beispiele für `*.pi` Dateien, die von Wing intern für die Erstellung von Auto-Vervollständigungsinformationen für Builtins verwendet werden, sind in dem Verzeichnis `resources/builtin-pi-files` innerhalb Ihrer Wing IDE Installation zu finden. Dieses veranschaulicht auch den oben beschriebenen Rückwärtsmechanismus der Versionsnummern.

### 4.19.1. Analyse-Cache

Der Source-Code-Analyser speichert Informationen über Dateien, die er kürzlich geprüft hat, unter `cache` in Ihrem **Verzeichnis der Benutzereinstellungen**.

Die Größe des Cache-Speichers kann mit der Einstellung **Maximale Cache-Größe** kontrolliert werden. Wing bringt allerdings keine so gute Leistung, wenn der für den Cache-Speicher verfügbare Raum kleiner ist als der Raum, der für die Source-Analyse-Informationen eines einzelnen Projekts benötigt wird. Wenn Sie extreme Verlangsamungen bemerken, erhöhen Sie entweder die Größe des Cache-Speichers oder deaktivieren Sie ihn vollständig, indem Sie seine Größe auf 0 setzen.

Wenn der Cache-Speicher von mehr als einem Computer verwendet wird, versichern Sie sich, dass die Uhren der beiden Computer synchronisiert sind. Der Cache-Mechanismus verwendet Zeitstempel und kann verwirrt werden, wenn dies nicht gemacht wird.

Der Analyse-Cache kann in seiner Gesamtheit ohne negative Auswirkungen entfernt werden.

# Source-Code-Browser

Der Source-Code-Browser, der nur in Wing IDE Professional und höheren Produktversionen verfügbar ist, dient als Index zu Ihrem Source-Code und unterstützt die Prüfung von Python-Source-Code-Sammlungen, entweder von einem modulorientierten oder Klassenhierarchie-orientierten Standpunkt.

**Source-Analyse im Hintergrund**
Wing IDE's Source-Code-Analyser wird vom Öffnen Ihres Projektes an solange im Hintergrund laufen bis alle Dateien analysiert sind. Sie werden diesen Overhead während der ersten 5 bis 30 Sekunden, nachdem Sie Ihr Projekt geöffnet haben, bemerken, abhängig von der Größe Ihrer Source-Basis. Bis die Analyse abgeschlossen ist, beinhaltet die klassenorientierte Ansicht innerhalb des Browser-Fensters nur solche Klassen, die analysiert wurden. Die Liste wird aktualisiert, sobald mehr Code analysiert ist.

## 5.1. Wahlmöglichkeiten für die Anzeige

Der Source-Code-Browser bietet drei Möglichkeiten, mit denen Sie auf die Sammlung Ihres Source-Codes sehen können: Nach Modul, nach Klassenhierarchie oder mit einer flachen Liste aller Klassen in Ihrem Projekt. Diese werden ausgewählt, indem Sie das Optionsmenü im Browser verwenden.

### 5.1.1. Nach Modul anzeigen

Das Anzeigen nach Modul zeigt in alphabetischer Reihenfolge alle Python-Module an, die Sie in Ihrem Projekt platziert haben, sowie alle Module, die durch das Traversieren der Verzeichnisstruktur, die Ihre Projektdateien enthält (einschließlich aller Unterverzeichnisse), erreicht werden. Die folgenden Typen von Top-Level Einheiten werden in diesem Anzeigemodus verwendet:

- Pakete sind Verzeichnisse, die eine Anzahl von Dateien und eine spezielle Datei `__init__.py` enthalten. Die Datei enthält optional eine spezielle Variable `__all__`, die die Datei-Level Module, die Python automatisch importieren sollte, wenn ein Paket als Ganzes importiert wird, enthält. Lesen Sie die Python-Dokumentation für zusätzliche Informationen über das Erstellen von Paketen.
- In Ihrem Projekt gefundene Verzeichnisse, die nicht die notwendige `__init__.py` Datei enthalten, werden als 'Verzeichnis' anstatt als 'Paket' im Source-Browser-Fenster aufgelistet.
- Python-Dateien, die auf einem beliebigen Level gefunden werden, tragen die Bezeichnung 'Modul'.

Innerhalb von jedem Top-Level Paket, Verzeichnis oder Modul zeigt der Browser alle Untermodule, Unterverzeichnisse, Module und alle möglichen Python-Konstrukte an. Diese sind alle mit einem generellen Typ gekennzeichnet, einschließlich der folgenden Typen:

- **Variable** -- Eine Variable, die im Top-Level eines Python-Moduls definiert ist.
- **Funktion** -- Eine Funktion, die im Top-Level eines Python-Moduls definiert ist.
- **Klasse** -- Eine Objektklasse, die in Python-Source-Code gefunden wird.
- **Methode** -- Eine Klassenmethode.
- **Attribut** -- Ein Klassen- oder Instanzattribut.

### 5.1.2. Klassenhierarchie anzeigen

Wenn nach Klassenhierarchie angezeigt wird, ersetzt der Browser die hierarchische Baumansicht mit einer Liste aller im analysierten Code gefundenen Top-Level Klassen (solche ohne irgendwelche Parent-Klassen) in alphabetischer Reihenfolge.

In diesem Anzeigemodus ist die Struktur Ihrer Pakete, Verzeichnisse und Module auf dem Laufwerk vollständig von der Ansicht verborgen. Stattdessen wird die Hierarchie Ihrer Klassen angezeigt, beginnend mit Basisklassen und absteigend zu den abgeleiteten Klassen.

Innerhalb jeder Klasse sind zusätzlich zu einer Liste der abgeleiteten Klassen alle Methoden und Attribute für die Klasse angezeigt.

### 5.1.3. Alle Klassen anzeigen

Um Klassen leichter nach Namen zu finden, kann der Browser aufgefordert werden, eine Liste anzuzeigen, die alle gefundenen Python-Klassen umfasst. In diesem Fall werden alle Klassen (und nicht nur die Basisklassen) im Top-Level der hierarchischen Ansicht angezeigt.

Diese Ansicht ist ansonsten mit der Ansicht der Klassenhierarchie identisch.

## 5.2. Anzeigefilter

Verschiedene Optionen sind für das Filtern der Konstrukte, die vom Source-Code-Browser präsentiert werden, verfügbar. Diese Filter stehen im Popup-Menü `Optionen` am Anfang des Browsers zur Verfügung. Sie sind in zwei Hauptgruppen eingeteilt: (1) Konstruktbereich und -Source und (2) Konstrukttyp.

### 5.2.1. Bereich und Source-Code filtern

Die folgenden Unterscheidungen werden getroffen. Konstrukte in jeder Kategorie können als Gruppe angezeigt oder versteckt werden:

- **Öffentlich** -- Konstrukte, die für jeden Benutzer eines Moduls oder einer Instanz zugänglich sind. Dies sind Namen, die null führende Unterstriche haben, wie `Print()` oder `kMaxListLength`.
- **Privat** -- Konstrukte, die zu einem Modul oder einer Klasse privat sein sollen. Dies sind Namen, die zwei führende Unterstriche haben, wie `__ConstructNameList()` oder `__id_seed`. Python erzwingt in Klassenmethoden nur lokalen Zugriff für diese Konstrukte (siehe Python-Dokumentation für Einzelheiten).
- **Halb-Privat** -- Konstrukte, die nur für den Gebrauch innerhalb von verwandten Modulen oder von verwandten oder abgeleiten Klassen gedacht sind. Dies sind Namen, die einen führenden Unterstrich haben, wie `_NotifyError()` oder `_gMaxCount`. Python erzwingt die Verwendung dieser Konstrukte nicht, aber sie sind hilfreich beim Schreiben von sauberem, gut strukturiertem Code und werden in der Python-Sprachstilanleitung empfohlen.
- **Geerbt** -- Konstrukte, die von einer Superklasse geerbt sind.
- **Importiert** -- Konstrukte, die mit einer Importanweisung in ein Modul importiert werden.

### 5.2.2. Konstrukttyp filtern

Konstrukte im Fenster des Source-Code-Browsers können auch auf Basis ihres grundlegenden Typs innerhalb der Sprache angezeigt oder versteckt werden:

- **Klassen** -- In Python-Source definierte Klassen.
- **Methoden** -- Methoden, die innerhalb von Klassen definiert sind.
- **Attribute** -- Attribute (auch bekannt als 'Instanzvariablen') einer Klasse. Beachten Sie, dass diese entweder klassenweit oder pro Instanz sein können, abhängig davon, ob sie innerhalb eines Klassenbereiches oder nur innerhalb von Methoden einer Klasse definiert sind.
- **Funktionen** -- Nicht-Objekt Funktionen, die in Python-Source definiert sind (normalerweise auf dem Top-Level eines Moduls).
- **Variablen** -- Variablen, die irgendwo in einem Modul, einer Klasse, Funktion oder Methode (aber nicht einschließlich der Funktions- oder Methodenparameter) definiert sind.
- **Abgeleitete Klassen** -- Klassen, die von einer anderen Klasse absteigen; dies steuert, ob von einer Klasse abgeleitete Klassen innerhalb ihres Bereiches angezeigt werden.

## 5.3. Die Browser-Anzeige sortieren

In allen Ansichten kann das Anordnen von Konstrukten innerhalb eines Moduls oder einer Klasse mit dem Drop-Down-Menü `Optionen` im Browser gesteuert werden.

- **Alphabetisch** -- Zeigt alle Einträge innerhalb jedes erweiterten Teil des Baumes in alphabetischer Reihenfolge an, unabhängig vom Typ.
- **Nach Typ** -- Sortiert jeden erweiterten Teil des Baumes zuerst nach Konstrukttyp und dann alphabetisch.

Das Sortieren beeinflusst nicht die höchste Ebene der hierarchischen Listenansicht, welche immer alphabetisch ist.

## 5.4. Navigation der Ansichten

Klicken Sie auf die Baumanzeige, um Source-Code vom Browser aus zu steuern. Dies öffnet Source-Dateien zu der entsprechenden Position. Beachten Sie, dass die Source-Dateien, die auf diese Weise geöffnet werden, automatisch geschlossen werden, wenn Sie an einer anderen Stelle durchsuchen; es sei denn, die Dateien wurden bearbeitet oder das Stick-Pin Symbol in der oberen rechten Ecke ist angeklickt, um anzuzeigen, dass die Source-Datei geöffnet bleiben soll, bis sie vom Nutzer ausdrücklich geschlossen wird.

Ein rechter Mausklick auf Klassen wird ein Popup-Menü präsentieren, das alle geerbten Klassen umfasst und schnelles Durchlaufen durch die Klassenhierarchie ermöglicht.

## 5.5. Tastaturnavigation des Browsers

Sobald sie den Fokus hat, ist die Baumansicht des Browsers mit der Tastatur steuerbar, und zwar unter Verwendung der Pfeiltasten oben/unten, Bild oben/unten und Pos1/Ende. Verwenden Sie außerdem die rechte Pfeiltaste auf einem Parent, um ihn zu erweitern oder die linke Pfeiltaste, um einen Parent zusammenzuklappen.

Wenn Sie die Umschalttaste gedrückt halten während Sie die rechte Pfeiltaste drücken, wird unter dem Erweiterungspunkt rekursiv erweitert. Die rekursive Erweiterung wird für jede Operation auf fünf zusätzlichen Ebenen begrenzt, um schwierig zu erkennende, unendliche Rekursionen zu vermeiden.

Immer wenn eine Baumreihe markiert ist, wird das Drücken der Eingabe- oder Return-Taste die Source-Ansicht für das gewählte Symbol in einem separaten Fenster öffnen und den Punkt der Definition für dieses Symbol anzeigen.

# Debugger

Wing's Debugger stellt ein leistungsfähiges Werkzeugset für die schnelle Lokalisierung und Behebung von Fehlern in Python-Code bereit. Er unterstützt Haltepunkte, das Schreiten durch den Code, die Prüfung und Änderung von Stack- oder Moduldaten, Watchpoints, Ausdrucksbewertung und die Interaktion im Command-Shell-Stil mit dem angehaltenen Debug-Prozess.

Der Debugger ist um ein TCP/IP Client/Server-Design erstellt, welches das Starten Ihrer Anwendung nicht nur von Wing selbst, sondern auch extern, wie mit CGI-Skripten oder Code, der in einer eingebetteten Skripting-Einrichtung innerhalb einer größeren Anwendung läuft, unterstützt. Remote-Debuggen (Host zu Host) steht auch zur Verfügung.

Da der Debugger-Kern in optimiertem C geschrieben ist, ist der Debug-Overhead relativ niedrig. Sie sollten jedoch damit rechnen, dass Ihre Programme innerhalb des Debuggers etwa 50% langsamer laufen.

## 6.1. Schnellstart

Wing IDE kann zum Debuggen aller Arten von Python-Code verwendet werden, einschließlich Skripten und selbständigen Anwendungen, die mit pygtk, **wxPython**, Tkinter, **PyQt** und pygame geschrieben sind. Wing kann außerdem **Web-CGIs debuggen**, einschließlich solchen, die unter **mod_python**, **Zope**-Produkten und externen Methoden laufen, sowie Code, der in einem eingebetteten Python-Interpreter läuft.

Dieser Abschnitt beschreibt, wie Sie selbständige Skripte und Anwendungen, die innerhalb von Wing IDE gestartet werden können, debuggen. Wenn Sie Web-CGIs innerhalb des Web-Servers, Zope-Code oder eingebetteten Python-Skripten debuggen möchten, lesen Sie bitte die Abschnitte **Extern gestarteten Code debuggen** und **Remote-Debuggen**.

Vor dem Debuggen müssen Sie Python auf Ihrem System installieren, wenn es noch nicht vorhanden ist. Python ist unter www.python.org erhältlich.

Um Python-Code mit Wing zu debuggen, öffnen Sie die Python-Datei und wählen `Debuggen / Fortsetzen` aus dem Menü Debuggen. Dies wird zum ersten Haltepunkt, zur ersten unbehandelten Exception oder bis zum Ende des Debug-Programms ausführen. Wählen Sie stattdessen `In Funktion`, um bis zur ersten Zeile des Codes zu laufen.

**Unerwartete Exceptions während dem Debuggen**
Wing kann Exceptions berichten, die Sie normalerweise nicht sehen, wenn Sie Ihren Debug-Prozess ausführen. Das passiert, wenn Exceptions auftreten, die in C oder C++ Erweiterungsmodul-Code behandelt (oder gelöscht) werden. Wing ermittelt alle Exceptions, die nicht in Python-Code behandelt werden. Sie können im Werkzeug Exceptions das Kontrollkästchen `Diese Exception-Position ignorieren` anklicken, um wiederholte Berichte von so einer Exception zu vermeiden, wenn es nicht von Interesse ist.

Verwenden Sie das Werkzeug Debug-I/O, um die Ausgabe Ihres Programms anzusehen oder um Werte für die Eingabe in das Programm einzutragen. Wenn Ihr Programm von den Eigenschaften der Windows-Konsole oder einer bestimmten Linux/Unix-Shell abhängt, lesen Sie bitte den Abschnitt **Externe I/O-Konsolen** für zusätzliche Informationen.

In einigen Fällen müssen Sie auch einen `PYTHONPATH` und andere Umgebungswerte eingeben. Verwenden Sie dafür den Dialog Projekteigenschaften, der über das Menü Projekt zugänglich ist. Dieser kann auch verwendet werden, um zu bestimmen, welchen Python-Interpreter Sie für Ihrem Debug-Prozess nutzen möchten. Verwenden Sie dies, wenn Wing IDE Python auf Ihrem System nicht finden kann oder wenn Sie mehr als eine Python-Version installiert haben.

Um Haltepunkte zu setzen, klicken Sie einfach auf den am weitesten links gelegenen Teil des Rands neben dem Source-Code.

In Wing IDE Professional sind bedingte Haltepunkte sowie Haltepunkte, die für eine bestimmte Anzahl von Zeiten ignoriert werden im Menü Debuggen / Haltepunkt-Optionen verfügbar.

## 6.2. Bestimmung des Debug-Startpunktes

Normalerweise wird Wing das Debuggen in der Datei, die im vordersten Editor aktiv ist, starten. In Abhängigkeit von der Natur Ihres Projektes möchten Sie vielleicht eine Datei als den Standard-Startpunkt für das Debuggen bestimmen.

Um dies einzurichten, klicken Sie mit der rechten Maustaste auf eine Ihrer Python-Dateien im Projektmanagerfenster und wählen aus dem Popup-Menü die Option

`Als Haupt-Debug-Datei einstellen` oder verwenden Sie den Eintrag `Aktuelle als Haupt-Debug-Datei einstellen` aus dem Debug-Menü.

Diese Datei wird im Folgenden immer dann ausgeführt, wenn Sie den Debugger starten, außer wenn Sie die Option `Aktuelle Datei debuggen` aus dem Debug-Menü verwenden oder wenn Sie mit der rechten Maustaste auf einen Eintrag im Projektmanager klicken und den Eintrag `Ausgewählte Debuggen` aus dem Popup-Menü wählen.

Beachten Sie, dass der Pfad zur Haupt-Debug-Datei im Projektfenster rot markiert ist. Sie können den Standard-Startpunkt für das Debuggen mit dem Eintrag des Popup-Menüs `Haupt-Debug-Datei löschen` aufheben oder Sie verwenden den Menüpunkt `Haupt-Debug-Datei löschen`, der über das Projektmenü zugänglich ist.

Der für ein Projekt definierte Debug-Startpunkt wird auch von der Source-Code-Analyse-Maschine verwendet, um die Version des Python-Interpreters und des Pythonpfades, die für die Analyse verwendet werden, zu bestimmen. Eine Änderung dieses Wertes verursacht also, dass alle Source-Dateien in Ihrem Projekt komplett neu analysiert werden. Siehe Abschnitt **Source-Code-Analyse** für Einzelheiten.

## 6.3. Debug-Eigenschaften

In einigen Fällen müssen Sie die Projekt- und Pro-Datei-Eigenschaften im Projektmanager einstellen, bevor Sie Code debuggen können. Dies wird gemacht, um den Python-Interpreter, `PYTHONPATH`, Umgebungsvariablen, Parameter, das Startverzeichnis und andere Werte, die mit dem Debug-Prozess in Verbindung stehen, zu bestimmen. Einzelheiten finden Sie in den Abschnitten **Projektweite Eigenschaften** und **Pro-Datei-Eigenschaften**.

## 6.4. Haltepunkte setzen

Haltepunkte können im Source-Code gesetzt werden, indem Sie die Source-Datei öffnen und links von einer Source-Code-Zeile auf den Haltepunktrand klicken. Alternativ können das Menü Debuggen oder die Haltepunktsymbole aus der Werkzeugleiste verwendet werden, um Haltepunkte an der aktuellen Zeile des Source-Codes (wo der Einfügecursor oder die Markierung ist) zu setzen oder zu löschen.

In Wing IDE Professional stehen Ihnen die folgenden Arten von Haltepunkten zur Verfügung:

- **Regulär** -- Ein regulärer Haltepunkt veranlasst den Debugger immer dazu, an einer gegebenen Code-Zeile zu stoppen, immer wenn dieser Code erreicht wird.

- **Bedingt** -- Ein bedingter Haltepunkt enthält einen Ausdruck, der jedes Mal, wenn der Haltepunkt erreicht wird, bewertet wird. Der Debugger wird nur dann stoppen, wenn die Bedingung als `wahr` bewertet wird (jeder beliebige Wert, der nicht null, leer oder `None` ist, wie von Python definiert). Sie können die Bedingung von jedem bestehenden Haltpunkt mit dem Eintrag `Haltepunkt-Bedingungen bearbeiten...` aus dem Menü Debuggen und dem Untermenü Haltepunkt-Optionen bearbeiten.

**Tastaturkombinationen für den Haltepunktrand**
Das Klicken auf den Haltepunktrand wird wechseln, um einen regulären Haltepunkt einzufügen oder einen bestehenden Haltepunkt zu entfernen.
Sie können außerdem mit Umschalt-Klick einen bedingten Haltepunkt einfügen und mit Strg-Klick einen Haltepunkt einfügen und für diesen eine Ignorieranzahl einstellen. Wenn auf der Zeile bereits ein Haltepunkt zu finden ist, wird er mit Shift-Klick deaktiviert oder aktiviert, Strg-Klick wird eine Ignorieranzahl setzen und Umschalt-Strg-Klick wird die Haltepunkt-Bedingung einstellen oder bearbeiten.

Wenn die Haltepunkte definiert wurden, können Sie in vielerlei Hinsicht mit ihnen arbeiten, um ihr Verhalten zu ändern. Diese Operationen sind entweder als Menüeinträge im Debuggen-Menü oder als Symbole in der Werkzeugleiste verfügbar:

- **Ignorieren** -- Es ist möglich, für einen Haltepunkt eine Ignorieranzahl einzustellen. In diesem Fall wird der Haltepunkt für die gegebenen Male ignoriert und der Debugger wird nur an diesem Haltepunkt stoppen, wenn die zu ignorierende Anzahl überschritten wird. Die Ignorieranzahl wird mit jeder neuen Debug-Ausführung auf ihren ursprünglichen Wert zurückgesetzt.
- **Deaktivieren/Aktivieren** -- Haltepunkte können vorübergehend deaktiviert und nachfolgend neu aktiviert werden. Jeder deaktivierte Haltepunkt wird solange ignoriert, bis der Nutzer ihn wieder neu aktiviert.
- **Löschen** -- Einzelne Haltepunkte können ausgewählt und entfernt werden.
- **Alle löschen** -- Es gibt auch einen Menüpunkt und ein Symbol in der Werkzeugleiste, mit dem alle definierten Haltepunkte auf einmal gelöscht werden können.

## 6.5. Debuggen starten

Es gibt mehrere Wege, eine Debug-Sitzung innerhalb von Wing zu starten:

- Wählen Sie `Debuggen / Fortsetzen` aus dem Menü Debuggen oder klicken Sie auf das Symbol `Debuggen` in der Werkzeugleiste. Das wird die Haupt-Debug-Datei, wenn eine bestimmt ist (beschrieben im Abschnitt **Eine Haupt-Debug-Datei setzen**), oder andernfalls die im vordersten Editor-Fenster geöffnete Datei ausführen. Die Ausführung hält am ersten Haltepunkt oder der ersten Exception an oder stoppt nach der Prgrommbeendigung.

- Wählen Sie `In Funktion` aus dem Menü Debuggen oder klicken Sie auf das Symbol `In Funktion` in der Werkzeugleiste. Dies wird die Haupt-Debug-Datei, wenn eine bestimmt ist, oder andernfalls die im vordersten Editor-Fenster geöffnete Datei ausführen. Die Ausführung stoppt an der ersten Code-Zeile.
- Wählen Sie `Aktuelle Datei debuggen` aus dem Menü Debuggen oder `Ausgewählte Debuggen` aus dem Popup-Menü, das mit einem rechten Mausklick auf das Projektwerkzeug aufgeschlagen wird, um eine spezifische Datei auszuführen, unabhängig davon, ob für Ihr Projekt eine Haupt-Debug-Datei bestimmt wurde. Dies wird am ersten Haltepunkt oder an der ersten Exception stoppen oder es wird nach Beendigung des Programms angehalten.
- Wählen Sie `Gehe zum Cursor` aus dem Menü Debuggen. Dies wird die Haupt-Debug-Datei, wenn eine bestimmt ist, oder andernfalls die im vordersten Editor-Fenster geöffnete Datei ausführen. Die Ausführung wird solange fortgesetzt bis sie die im aktuellen Source-Text-Fenster markierte Zeile erreicht, bis sie auf einen Haltepunkt oder eine Exception trifft oder bis das Programm beendet ist.
- Verwenden Sie `Letzte Debuggen` aus dem Menü Debuggen, um eine kürzlich gedebuggte Datei auszuwählen. Dies wird am ersten Haltepunkt oder an der ersten Exception stoppen oder nach Beendigung des Programms anhalten.
- Verwenden Sie einen der Tastaturbefehle, die im Menü Debuggen zu finden sind. Im Emacs-Modus ist der Tastaturbefehl `Strg-C Strg-C` auch implementiert.

Zusätzliche Optionen bestehen für das Starten einer Debug-Sitzung von außerhalb von Wing IDE und für das Anhängen an einen bereits laufenden Debug-Prozess. Diese sind in den Abschnitten **Extern gestarteten Code debuggen** und **Anhängen** beschrieben.

Sobald ein Debug-Prozess gestartet wurde, sollte sich das Statuslicht in der oberen rechten Ecke des Werkzeuges Stack-Daten von rot auf eine andere Farbe ändern, wie in **Debugger-Status** beschrieben.

**Nicht standardisierte Python-Interpreter**
Wenn Sie versuchen, Ihren Debug-Prozess gegen eine nicht standardisierte Version von Python auszuführen, zum Beispiel eine, die mit geänderten Werten für `Py_TRACE_REFS` oder `WITH_CYCLE_GC` kompiliert wurde oder eine, die auf andere Weise verändert wurde, dann müssen Sie wahrscheinlich das Debugger-Kernmodul neu kompilieren. Bitte lesen Sie für zusätzliche Informationen den Abschnitt **Den Wing IDE Debugger vom Source-Code kompilieren**. Dies ist nur in Wing IDE Professional möglich, da Wing IDE Personal keinen Zugriff auf den Source-Code umfasst.

## 6.6. Debugger-Status

Die Debugger-Werkzeuge Stack-Daten, Beobachten und Debug-Test enthalten eine Statusanzeige, die sich rechts neben dem Popup-Menü für die Stack-Auswahl befindet. Die Statusanzeige kann verwendet werden, um den Zustand des Debuggers folgendermaßen zu bestimmen:

- **Reines Rot** -- Es existiert kein Debug-Prozess und der Debugger hört nicht auf Verbindungen.
- **Rot mit Schrägstrich** -- Es existiert kein Debug-Prozess, aber der Debugger hört auf Verbindungen von extern gestarteten Prozessen.
- **Gelb** -- Ein Debug-Prozess ist angehängt und wird ausgeführt oder er ist beim Anhängen.
- **Grün** -- Ein Debug-Prozess ist angehängt und an einem Haltepunkt oder einer Exception angehalten oder gestoppt.

Wenn Sie mit der Maus über die Statusanzeige fahren, wird ein Werkzeug-Tipp angezeigt, der den Debugger-Status beschreibt.

Der aktuelle Status des Debuggers ist außerdem im Nachrichtenwerkzeug des IDE's in der Debugger-Statusgruppe aufgelistet.

## 6.7. Ablaufsteuerung

Wenn der Debugger einmal läuft, sind die folgenden Befehle zur Kontrolle der weiteren Ausführung des Debug-Programms von Wing verfügbar. Diese können über die Werkzeugleiste oder das Menü Debuggen erreicht werden:

- Ein frei-laufendes Debug-Programm kann jederzeit mit der Option `Anhalten` aus dem Menü Debuggen oder mit der Schaltfläche Anhalten aus der Werkzeugleiste angehalten werden. Dies wird am aktuellen Punkt der Ausführung des Debug-Programms anhalten.
- Während einer Debug-Sitzung kann jederzeit der Menüeintrag oder das Werkzeug `Debuggen Stoppen` verwendet werden, um die Beendigung des Debug-Programms zu erzwingen.

  Diese Option ist standardmäßig deaktiviert, wenn der aktuelle Prozess außerhalb von Wing gestartet wurde. Sie kann für alle lokalen Prozesse mit der Einstellung **Externe Löschen aktivieren** aktiviert werden.

Wenn an einer gegebenen Code-Zeile gestoppt wurde, kann die Ausführung mit dem Menü Debuggen oder der Werkzeugleiste wie folgt kontrolliert werden:

- **Über Funktion** schreitet über eine einzelne Zeile von Python-Code.
- **In Funktion** wird versuchen, in die nächste ausgeführte Funktion in der aktuellen Code-Zeile zu gehen. Wenn dort keine Funktion oder Methode ist, in die hineingegangen werden kann, dann verhält sich dieser Befehl wie Aus Funktion.
- **Aus Funktion** wird die Ausführung der aktuellen Funktion oder Methode abschließen und an der ersten Anweisung, die nach der Rückkehr von der aktuellen Funktion oder Methode angetroffen wird, stoppen.
- **Fortsetzen** wird die Ausführung bis zum nächsten Haltepunkt, zur nächsten Exception oder zum Programmende fortsetzen.
- **Gehe zum Cursor** wird zur Stelle des Cursors im vordersten Editor oder zum nächsten Haltepunkt, zur nächsten Exception oder zum Programmende gehen.
- **Anhängen** und **Entfernen** (nur in Wing IDE Professional) kann verwendet werden, um den Debugger zwischen verschiedenen Debug-Prozessen zu wechseln. Dies ist für fortgeschrittene Nutzer und ist im Abschnitt **Anhängen und Entfernen** detailliert beschrieben.

## 6.8. Stack anzeigen

Immer wenn das Debug-Programm an einem Haltepunkt oder während dem manuellen Schreiten anhält, wird der aktuelle Stack im Popup-Menü am Anfang des Werkzeuges Stack-Daten angezeigt. Dies zeigt alle Stack-Frames, die zwischen der Anforderung des

Programms und der aktuellen Ausführungsposition angetroffen werden, an. Die äußeren Stack-Frames sind in der Liste weiter oben.

Beachten Sie, dass der angezeigte Stack eine Verkettung von allen gesehenen Python Stack-Frames ist und Unstetigkeiten enthalten kann, wenn Ihr Code C/C++ oder anderen nicht-Python-Code aufruft, der im Gegenzug in Python zurückruft. In diesem Fall werden die C/C++ Stack-Frames fehlen, aber die insgesamte Reihenfolge und der Fluss der Anforderung sollten von denjenigen Stack-Frames, die sichtbar sind, offensichtlich sein.

Wenn der Debugger an einen Haltepunkt oder eine Exception geht oder dort stoppt, wählt er standardmäßig den innersten Stack-Frame aus.

Um andere Stack-Frames weiter oben oder unter im Stack zu besuchen, wählen Sie sie aus dem Popup-Menü der Stack-Daten aus, verwenden die Einträge `Aufwärts Stack` und `Abwärts Stack` aus dem Menü Debuggen oder klicken auf die Symbole Aufwärts/Abwärts in der Werkzeugleiste.

Wenn Sie Stack-Frames wechseln, werden die Variablenansichten entsprechend geändert und die aktuelle Code-Zeile an diesem Stack-Frame wird in einem Editor-Fenster angezeigt.

Beachten Sie, dass der aktuelle Stack-Frame auch verwendet wird, um den Bewertungskontext in den Werkzeugen Debug-Test und Beobachten zu steuern.

## 6.9. Debug-Daten anzeigen

Der Wing IDE Debugger stellt mehrere Möglichkeiten bereit, mit denen Sie die Daten Ihres Debug-Programms ansehen können:

(1) Durch die Prüfung von Lokalen und Globalen unter Verwendung des Werkzeuges Stack-Daten. Dieser Bereich zeigt Werte für den gegenwärtig gewählten Stack-Frame an.

(2) Durch das Durchsuchen von Werten in allen geladenen Modulen (wie von `sys.modules` bestimmt), unter Verwendung des Werkzeuges Module.

(3) Durch das Beobachten spezifischer Werte von einer der oben genannten Ansichten (klicken Sie mit der rechten Maustaste auf Werte, um sie zum Werkzeug Beobachten hinzuzufügen).

(4) Durch das Eingeben von Ausdrücken in das Beobachten-Werkzeug.

**Auf Anfrage abgerufene Werte**
Die von Wing angezeigten Variablendaten werden vom Debug-Server schnell abgerufen während Sie navigieren. Aus diesem Grund können Sie eine kurze Verzögerung bemerken, wenn eine Änderung in einer Erweiterung oder einem Stack-Frame zu einer großen Datenübertragung führt.
Aus dem gleichen Grund können große Mengen von Debug-Daten, die auf dem Bildschirm sichtbar bleiben, das Schreiten durch den Code verlangsamen.

### 6.9.1. Ansicht der Stack-Daten

Das Debugger-Werkzeug Stack-Daten enthält ein Popup-Menü für den Zugriff auf den aktuellen Debug-Stack, einen Baumansichtsbereich für das Durchsuchen von Variablendaten in Lokalen und Globalen und einen Textansichtsbereich für die Prüfung von großen Datenwerten, die in der Baumansicht abgeschnitten sind.

Einfache Werte, wie Strings und Zahlen, und Werte mit einer kurzen String-Ansicht werden in der Wertspalte des Baumansichtbereichs angezeigt.

Strings sind immer in `""` (Anführungszeichen) eingebettet. Jeder Wert außerhalb der Anführungszeichen ist eine Zahl oder eine intern definierte Konstante, wie `None` oder `Ellipsis`.

Ganzzahlen können entweder als dezimal, hexadezimal oder oktal angezeigt werden, was mit der Einstellung **Anzeigemodus für Ganzzahlen** geregelt werden kann.

Komplexe Werte, wie Instanzen, Lists und Dictionaries, werden in eckigen Klammern und mit einer Speicheradresse dargestellt (zum Beispiel `<dict 0x80ce388>`) und können durch Klicken auf die Erweiterungsanzeige in der Spalte Variable erweitert werden. Die Speicheradresse identifiziert das Konstrukt eindeutig. Wenn Sie die gleiche Adresse an zwei Stellen sehen, dann betrachten Sie zwei Objektverweise zu der gleichen Instanz.

Die in den Bereich einer Klasse gehörenden Werte, die innerhalb einer Instanz gesehen werden, sind kursiv angezeigt.

Nach der Erweiterung komplexer Ansichten wird die Position oder der Name von jedem Untereintrag in der Spalte Variable angezeigt und der Wert von jedem Eintrag (möglicherweise auch komplexe Werte) werden in der Spalte Wert angezeigt. Verschachtelte, komplexe Werte können unbestimmt erweitert werden, selbst wenn dies zu einem Traversal der Zyklen der Objektverweise führt.

Wenn Sie einen Eintrag einmal erweitern, wird der Debugger diesen Eintrag weiterhin als erweitert darstellen, selbst nachdem Sie weiter gehen oder die Debug-Sitzung neu starten. Der Erweiterungsstatus wird für die Dauer Ihrer Wing IDE Sitzung gespeichert.

Wenn der Debugger auf einen langen String trifft, wird dies in der Spalte Wert durch das Voranstellen von ... vor dem abgeschnittenen String angezeigt. In diesen Fällen kann der vollständige Wert des Strings im Textansichtsbereich unten im Stack-Daten-Werkzeug angesehen werden, nachdem in der Baumansicht auf den abgeschnittenen String geklickt wurde.

> **Unlesbare Daten**
> Einige Datentypen, wie zum Beispiel solche, die nur innerhalb von C/C++ Code definiert sind oder solche, die bestimmte Internals der Python-Sprache beinhalten, können nicht über das Netzwerk übertragen werden. Diese sind mit Werteinträgen in der Form `<opaque 0x80ce784>` bezeichnet und können nicht weiter erweitert werden. In Wing IDE Professional können Sie allerdings den **Debug-Test** verwenden, um auf sie zuzugreifen (versuchen Sie zum Beispiel `dir(value)` einzutippen).

### 6.9.1.1. Optionen des Popup-Menüs

Ein rechter Mausklick auf die Oberfläche der Stack-Datenansicht schlägt ein Popup-Menü mit Optionen für die Navigation von Datenstrukturen auf:

- **Mehr Erweitern** -- Wenn ein komplexer Datenwert ausgewählt ist, wird dieser Menüeintrag eine zusätzliche Ebene in dem komplexen Wert erweitern. Da dies eine potentiell große Anzahl von Werten erweitert, werden Sie wahrscheinlich eine Verzögerung bemerken, bis dieser Vorgang abgeschlossen ist.
- **Mehr Zusammenklappen** -- Wenn ein komplexer Datenwert ausgewählt ist, wird dieser Menüeintrag dessen Anzeige um eine zusätzliche Ebene zusammenklappen.
- **Nach ... beobachten** -- Diese Punkte können verwendet werden, um einen Debug-Datenwert über die Zeit zu beobachten, wie in **Werte verfolgen** beschrieben.
- **Neuladen erzwingen** -- Dies zwingt Wing IDE, den angezeigten Wert vom Debug-Prozess neuzuladen. Dies ist in Fällen hilfreich, in denen Wing einen Bewertungsfehler anzeigt oder wenn das Debug-Programm Instanzen enthält, die `__repr__` oder ähnliche besondere Methoden in einer Art implementieren, die verursacht, dass sich der Wert ändert, wenn er der wiederholten Bewertung unterliegt.

### 6.9.1.2. Anzeige von Werten filtern

Es gibt eine Vielzahl von Möglichkeiten, mit denen die Anzeigen für Variablen konfiguriert werden können:

- Wing lässt Sie den Variablenanzeigebereich entfernen, indem Sie alle Werte nach Typ auslassen und Variablen- oder Dictionary-Schlüssel nach Namen auslassen. Dies wird durch das Setzen der zwei Einstellungen **Typen auslassen** und **Namen auslassen** vorgenommen.

- Sie können Wing auch mitteilen, dass es das Testen von bestimmten Werten nach Datentyp vermeiden soll. Dies ist nützlich, um den Versuch der Erweiterung von Datenwerten, die in fehlerhaften Erweiterungsmodulen definiert sind, zu vermeiden. Diese können sonst zum Abstürzen des Debug-Prozesses führen, wenn der Debugger Code ausführt, der normalerweise nicht ausgeführt wird. Um zu vermeidende Werte hinzuzufügen, setzen Sie die Einstellung **Nicht Erweitern**.

- Wing stellt Kontrollmöglichkeiten mittels Größenschwellen bereit, über denen Werte als zu groß angesehen werden, um vom Debug-Prozess in den Variablenanzeigebereich verschoben zu werden. Werte, die zu groß sind, werden im Variablenanzeigebreich als `Riesig` kommentiert und können nicht weiter erweitert werden. Die Schwellen für Datengrößen werden mit den Einstellungen **Große Listenschwelle** und **Große Stringschwelle** gesteuert.

- Standardmäßig wird Wing kleine Einträge auf einer einzelnen Zeile in den Variablenanzeigebereichen anzeigen, selbst wenn es komplexe Typen, wie Listen und Maps, sind. Die dafür verwendete Größenschwelle wird mit der Einstellung **Zeilenschwelle** geregelt. Wenn Sie möchten, dass alle Werte einheitlich angezeigt werden, sollte die Einstellung auf `0` gesetzt werden.

### 6.9.2. Werte verfolgen

Wing kann Debug-Datenwerte beobachten, indem es eine Vielzahl von Techniken für das Verfolgen eines Wertes über die Zeit verwendet. In den meisten Fällen wird das Beobachten eines Wertes durch einen Rechtsklick auf den Wert innerhalb der Baumansicht und die Auswahl eines Eintrages aus dem Beobachtungsmenü gestartet. Der Wert wird dann zu der Liste im Beobachten-Werkzeug hinzugefügt und mit einer der folgenden Methoden verfolgt:

- **Nach symbolischem Pfad** - Der Debugger betrachtet den symbolischen Pfad von `locals()` oder `globals()` für den gegenwärtig gewählten Stack-Frame und versucht diesen Pfad immer dann neu zu beurteilen, wenn sich der Wert geändert haben kann. Wenn Sie zum Beispiel eine Dictionary-Variable mit dem Namen `testdict` in einer Funktion definieren und einen Wert `testdict[1] = 'test'` setzen, dann würde der beobachtete Wert für `testdict[1]` jeden Wert für diesen

Slot von `testdict` anzeigen, selbst wenn Sie `testdict` löschen und neu erstellen. Mit anderen Worten: Die Datenverfolgung ist vom Bestehen von irgendwelchen Objektinstanzen im Datenpfad unabhängig.

- **Nach direktem Objektverweis** - Der Debugger verwendet den Objektverweis auf den gewählten Wert, um ihn zu verfolgen. Wenn Sie diesen Modus mit `testdict` als Ganzes verwenden, dann würde es die Inhalte von diesem Dictionary verfolgen, so lange wie dieses existiert. Wenn Sie die Variable `testdict` einem anderen Wert neu zuweisen würden, dann würde Ihre vergrößerte Ansicht noch die Inhalte der ursprünglichen Dictionary-Instanz (wenn sie noch existiert) anzeigen, anstatt des neuen Wertes der Variable `testdict`. Mit anderen Worten: Der symbolische Pfad zum Wert wird vollständig ignoriert und zur Verfolgung des Wertes wird nur die Instanzidentität verwendet. Da es sinnlos ist, unveränderliche Typen auf diese Weise zu verfolgen, ist diese Option deaktiviert oder aktiviert, entsprechend dem Wert, den Sie für die Vergrößerung in einem separaten Fenster ausgewählt haben.
- **Nach Parent-Verweis und Slot** - Der Debugger verwendet den Objektverweis auf den Parent des gewählten Daten-Slots und verwendet eine symbolische Darstellung des Slots innerhalb des Parent, um zu bestimmen, wo nach irgendwelchen Wertaktualisierungen zu suchen ist. Das bedeutet, dass die Neuzuweisung der Variablen, die auf den Parent zeigt, nicht ändert, was in der vergrößerten Ansicht angezeigt wird; nur die Neuzuweisung des gewählten Slots ändert, was vom Debugger angezeigt wird.
- **Nach Modul-Slot** - Dies ist nur für Werte innerhalb von Modulen verfügbar, wie `string`, `sys.path` oder `os.environ`. Der Debugger verwendet den Modulnamen, um das Modul in `sys.modules` herauszusuchen und verweist auf den Wert durch den symbolischen Pfad. Jegliche Änderungen in dem Wert, sogar über Modulneuladungen, werden in der Beobachten-Ansicht reflektiert.

Für alle von diesen gilt: Wenn der Wert nicht bewertet werden kann, weil er nicht existiert, zeigt der Debugger `<undefiniert>` an. Dies passiert, wenn der letzte Objektverweis zu einem Wert, der mit einem Verweis verfolgt wird, abgelegt wird oder wenn ein gewählter symbolischer Pfad undefiniert ist oder nicht bewertet werden kann.

Das Beobachten-Werkzeug speichert Watchpoints über die Debug-Sitzungen, außer denen, die von einem Objektverweis, der den Debug-Prozess nicht überlebt, Gebrauch machen.

### 6.9.3. Ausdrücke bewerten

Das Beobachten-Werkzeug des Debuggers kann auch verwendet werden, um den Wert von Ausdrücken, die mit der Tastatur eingegeben wurden, anzusehen. Diese können

eingegeben werden, indem auf jede beliebige Zelle im Anzeigebaum des Beobachtungsmanagers geklickt wird und der gewünschte Ausdruck in der Spalte Variable bearbeitet oder eingegeben wird. Drücken Sie die Eingabetaste, um die Bearbeitung zu beenden.

Es können nur Ausdrücke, die zu einem Wert bewerten, eingegeben werden. Andere Anweisungen, wie Zuweisungen von Variablen, Importanweisungen und Sprachkonstrukte, werden mit einer Fehlermeldung abgewiesen. Diese können nur mit Verwendung des **Debug-Tests** ausgeführt werden.

Ausdrücke werden im Kontext des aktuellen Debug-Stack-Frames bewertet, d.h. diese Funktion ist nur verfügbar, wenn das Debug-Programm an einem Haltepunkt oder einer Exception angehalten oder gestoppt wurde. Das bedeutet auch, dass sich der Wert des gleichen geschriebenen Ausdrucks verändern kann, wenn Sie den Call-Stack im Haupt-Debugger-Fenster nach oben und unten verschieben.

In Fällen, in denen das Bewerten eines Ausdrucks zur Änderung des Wertes von lokalen oder globalen Variablen führt, wird Ihr Debug-Programm in diesem geänderten Kontext fortfahren. Immer wenn ein Wert aufgrund einer Ausdrucksbewertung geändert wird, wird der aktualisierte Wert in alle sichtbaren Anzeigenbereiche der Debugger-Variablen verbreitet, weil Wing IDE alle angezeigten Datenwerte nach der Bewertung von jedem Ausdruck neu abruft. Da es jedoch sein kann, dass Sie diese Änderungen nicht bemerken, ist Vorsicht geboten, um ungewünschte Nebeneffekte im Debug-Prozess zu vermeiden.

Beachten Sie, dass Haltepunkte niemals aufgrund der Ausdrucksbewertung erreicht werden und dass Exceptions, die angetroffen werden, nicht berichtet werden. Wenn Sie einen Ausdruck debuggen müssen, verwenden Sie den **Debug-Test**, bei dem Exceptions berichtet werden.

### 6.9.4. Probleme bei der Behandlung von Werten

Der Wing Debugger versucht, Debug-Daten so sanft wie möglich zu behandeln, um das Eintreten von sehr langen Berechnungen oder das Auslösen von Fehlern im Debug-Prozess während dem Packen von Debug-Daten für die Übertragen zu vermeiden. Trotzdem können nicht alle Debug-Daten in der Anzeige angezeigt sein. Dieser Abschnitt beschreibt alle Gründe, warum dies passieren kann:

- **Wing kann bei der Behandlung eines Wertes abschalten** -- Große Datenwerte können den Debug-Server-Prozess während dem Packen aufhängen. Wing versucht dies zu vermeiden, indem es die Größe eines Objektes vor dem Packen sorgfältig testet. In einigen Fällen funktioniert das nicht und Wing wird auf die Daten für die Dauer, die in der Einstellung **Netzwerkabschaltung** festgelegt ist, warten und wird dann den Variablenwert als `<Netzwerkabschaltung während der Bewertung>` anzeigen.

- **Wing kann auf Werte treffen, die für die Verarbeitung zu groß sind** -- Wing wird große Sequenzen, Bereiche oder Strings, die die von den Einstellungen **Große Listenschwelle** und **Große Stringschwelle** festgelegten Größenlimits übersteigen, nicht verpacken und übertragen. In der Debugger-Anzeige werden übergroße Sequenzen und Bereiche als `riesig` kommentiert und `<abgeschnitten>` ist großen, abgeschnittenen Strings vorangestellt.

  Erhöhen Sie die Werte der Schwelleneinstellungen, um dies zu vermeiden; seien Sie aber auf längere Datenübertragungszeiten vorbereitet. Beachten Sie, dass eine zu hohe Einstellung dieser Werte den Debugger zum Abschalten veranlassen wird, wenn der Wert **Netzwerkabschaltung** nicht auch erhöht wird.

  Eine Alternative, die in Wing IDE Professional für das Ansehen großer Datenwerte zur Verfügung steht, besteht darin, Ausdrücke in das **Beobachten-Werkzeug** oder den **Debug-Test** einzugeben, um Unterteile der Daten zu sehen, anstatt den ganzen Top-Level Teil des Wertes zu übertragen.

- **Wing kann während der Datenbearbeitung auf Fehler treffen** -- Da Wing während dem Packen von Debug-Daten Zuweisungen und Vergleiche vornimmt, und weil es Debug-Daten in String-Form umwandelt, kann es besondere Methoden, wie `__cmp__` und `__str__` in Ihrem Code ausführen. Wenn dieser Code Fehler enthält, kann der Debugger diese Fehler manchmal aufdecken, die Sie ansonsten nicht sehen würden.

  Im schlimmsten Fall, der allerdings nur selten eintritt, wird der Debug-Prozess abstürzen, wenn beschädigter C oder C++ Erweiterungsmodul-Code aufgerufen wird. In diesem Fall wird die Debug-Sitzung beendet.

  Allgemein bekannter, aber trotzdem noch selten, sind Fälle, in denen Wing während der Bearbeitung eines Debug-Datenwertes auf eine unerwartete Python-Exception trifft. Wenn dies passiert, zeigt Wing den Wert als `<Fehler bei Wertbearbeitung>` an.

  Diese Fehler werden im Exceptions-Werkzeug nicht als normale Programmfehler berichtet. Zusätzliche Ausgabe, welche die aufgetretene Exception enthalten kann, kann jedoch durch das Setzen der Einstellung **Protokolldatei der Internals debuggen** erhalten werden.

Wing merkt sich Fehler, auf die es in Debug-Daten trifft, und speichert diese in der Projektdatei. Diese Werte werden während nachfolgendem Debuggen nicht neu abgerufen, selbst wenn Wing beendet und neu gestartet wird.

Um dieses Verhalten für einen einzelnen Wert außer Kraft zu setzen, verwenden Sie den Menüpunkt `Neuladen erzwingen` aus dem mit einem rechten Mausklick aufzuschlagenden Popup-Menü eines Variablenbereiches in Baumansicht.

Verwenden Sie den Eintrag `Gespeicherte Wertefehler löschen` aus dem Menü `Debuggen`, um die Liste aller vorher angetroffenen Fehler zu löschen, so dass alle Werte neu geladen werden. Dies funktioniert nur für die Liste der Fehler, die in der aktuellen Debug-Datei bekannt sind, wenn eine Debug-Sitzung aktiv ist oder für die Haupt-Debug-Datei, wenn vorhanden, wenn kein Debug-Prozess läuft.

## 6.10. Interaktiver Debug-Test

Der Debug-Test funktioniert wie die Python-Shell für das Bewerten und Ausführen von beliebigem Python-Code im Kontext eines Debug-Programms. Dies funktioniert auf dem aktuellen Debug-Stack-Frame und ist folglich nur verfügbar, wenn das Debug-Programm angehalten ist.

Sie können viele von Wing's Source-Editor-Befehlen und Tastaturkombinationen innerhalb des Debug-Tests verwenden und Sie können die Pfeiltasten nach oben/unten nutzen, um eine Historie von kürzlich eingegebenen Befehlen zu durchlaufen.

Wenn von Ihnen eingetippte Befehle irgendwelche lokalen, Instanz- oder globalen Datenwerte ändern, wenn sie verursachen, dass Module geladen oder entladen werden, wenn sie Umgebungsvariablen setzen oder die Ausführungsumgebung anderweitig ändern, wird Ihr Debug-Programm in diesem geänderten Zustand fortfahren. Alle sichtbaren Anzeigen der Variablenansicht werden auch nach jeder Zeile, die in den Debug-Test eingegeben wird, aktualisiert, um alle Änderungen, die durch Ihre Befehle veranlasst werden, widerzuspiegeln. Da Sie diese Änderungen vielleicht nicht bemerken, ist Vorsicht geboten, um die Erschaffung von ungewünschten Nebeneffekten in dem ausführenden Debug-Programm zu vermeiden.

Eine Beschränkung ist, dass private Instanzvariablen, denen ein doppelter Unterstrich vorangestellt ist (wie `self.__my_var`), innerhalb der interaktiven Shell nicht direkt geprüft oder geändert werden können. Python wird berichten, dass das Attribut nicht definiert ist, weil es intern dem Klassennamen, in dem die private Variable zu finden ist, vorangestellt ist. Diese können einfach mit dem Werkzeug Stack-Daten angesehen werden. Alternativ können Sie die vollständig qualifizierte Form des privaten Instanz-Variablennamens verwenden: Um zum Beispiel auf `__my_var` in einer Instanz der Klasse `myclass` zuzugreifen, verwenden Sie `self._myclass__my_var` im Debug-Test.

Beachten Sie, dass in dieser Wing-Version Haltepunkte niemals als Folge von im Debug-Test eingegebenen Einträgen erreicht werden und dass alle möglichen Exceptions nur nach der Tatsache berichtet werden. Das bedeutet, dass Aktivität im Debug-Test keine Auswirkungen auf die Debug-Ausführungsposition oder den Stack hat, selbst wenn in einigen Fällen eine Exception-Position im Source-Code angezeigt werden kann.

Die Einstellung **Source von Werkzeugen aufschlagen** kann verwendet werden, um zu

bestimmen, ob Source-Code-Fenster aufgeschlagen werden, wenn Exceptions im Debug-Test auftreten.

## 6.11. Interaktive Python-Shell

Eine Python-Shell wird für die Ausführung von Befehlen und die Bewertung von Ausdrücken außerhalb Ihres Debug-Programms bereitgestellt.

Da diese Shell einen separaten Python-Prozess, der von Ihrem Debug-Prozess unabhängig ist, ausführt, ist sie immer aktiviert und funktioniert ohne Rücksicht auf den Status eines laufenden Debug-Prozesses.

Die Python-Shell läuft immer mit der gleichen Python-Version, wie die, die auch für Ihren Debug-Prozess verwendet wird. Dies wird genauer im Abschnitt **Debug-Eigenschaften** beschrieben.

Um den Status einer Python-Shell aufzuheben, drücken Sie auf die Schaltfläche `Neue Sitzung`. Dies wird den externen Python-Prozess beenden und ihn neu starten, also den Status der Shell aufheben und neu einstellen.

## 6.12. Exceptions verwalten

Wing's Debugger versucht, von Ihrem Debug-Prozess angetroffene, unbehandelte Exceptions zu erkennen, und wird Ihnen diese sofort berichten. Dies ermöglicht Ihnen, den Programmzustand, der zu der Exception geführt hat, anzusehen und erlaubt, durch nachfolgend erreichte `finally` Clauses zu schreiten. Dies wird erreicht, indem im Stack nach Exception-Handlern gesucht wird, die in Python geschrieben sind, und indem nur Exceptions berichtet werden, für die es keinen passenden Handler gibt.

Diese Technik funktioniert gut mit wxPython, PyGTK und in fast allen anderen Codes, in denen unerwartete Exceptions entweder zur Programmbeendigung führen oder von Catch-all Exception-Handlern, die in C/C++ Erweiterungsmodul-Code geschrieben sind, verarbeitet werden.

Bei Code mit in Python geschriebenen Catch-all Exceptions kann Wing manchmal daran scheitern, unerwartete Exceptions zu berichten, es sei denn, diese Handler werden wie im Abschnitt **Fehler beim Stoppen an Exceptions** umgeschrieben.

In einigen Fällen kann es vorkommen, dass Wing's Erkennungsmechanismus für unbehandelte Exceptions normal verarbeitete Exceptions, die außerhalb des Debuggers nicht zu sehen sind, berichtet. Dies passiert, wenn die Exceptions im C/C++

Erweiterungsmodul-Code verarbeitet werden. Sie können Wing trainieren, diese Exceptions zu ignorieren, wenn Sie das Kontrollkästchen `Diese Exception-Position ignorieren` im Debugger-Werkzeug `Exceptions` anklicken. Ignorierte Exceptions werden immer noch berichtet, wenn sie tatsächlich zur Beendigung des Programms führen. Ihre Auswahl wird in der Projektdatei gespeichert, so dass Sie sie nur einmal vornehmen müssen. Sie können jederzeit den Menüpunkt `Ignorierte Exceptions löschen` im Menü `Debuggen` verwenden, um die Ignorierliste zu bereinigen.

Die Einstellung **Berichten von Exceptions** kann verwendet werden, um Wing's vorhersagenden Exception-Handler auszuschalten, und zwar zugunsten anderer Möglichkeiten, die darüber entscheiden, welche Exceptions während der Laufzeit berichtet werden sollten. Die folgenden Auswahlmöglichkeiten für das Berichten von Exceptions stehen zur Verfügung:

- **Sofort, wenn scheinbar unbehandelt** -- Dies ist die oben beschriebene Voreinstellung. Der Debugger wird sofort an Exceptions anhalten, wenn diese angetroffen werden, aber nur, wenn für diese Exception kein Handler gefunden wird. Falsche Positive im Werkzeug `Exceptions` können ignoriert werden.

- **Immer sofort** -- Der Debugger wird sofort an jeder einzelnen Exception anhalten, wenn diese angetroffen wird. In dem meisten Code wird dies sehr oft passieren, da Exceptions intern für die Verarbeitung normaler, akzeptabler Laufzeitbedingungen verwendet werden können.

- **Beim Beenden des Debug-Prozesses** -- In diesem Fall wird der Debugger an Exceptions, die tatsächlich zur Beendigung des Prozesses führen, stoppen und diese berichten. Dies passiert genau bevor oder manchmal genau nachdem der Prozess beendet wurde. Die Exception wird auch in `stderr` gespeichert, so wie es beim Ausführen außerhalb des Debuggers der Fall wäre.

  Wenn Sie mit einem **Extern gestarteten Debug-Prozess** arbeiten, kann es passieren, dass der Modus `Beim Beenden des Debug-Prozesses` nicht in der Lage ist, den Debug-Prozess vor dem Beenden zu stoppen. In einigen Fällen kann er sogar daran scheitern, überhaupt eine Rückverfolgung nach dem Beenden anzuzeigen (außer der im Debug-Prozess in `stderr` gespeicherten Daten).

  Gleichermaßen wird auch beim Arbeiten mit wxPython, PyGTK und ähnlichen Umgebungen, die einen Catch-all Exception-Hander in C/C++ Code umfassen, der Modus `Beim Beenden des Debug-Prozesses` daran scheitern, unerwartete Exceptions, die während der Hauptschleife auftreten, zu berichten, da diese Exceptions nicht zur Beendigung des Prozesses führen.

  Für diese beiden Fällen empfehlen wir Ihnen, den Exception-Berichtmodus `Sofort, wenn scheinbar unbehandelt` zu verwenden.

## 6.13. Debug-Prozess-I/O

Während Sie unter dem Wing Debugger ausführen, wird der gesamte Verkehr zu und von Python `stdin` und `stdout` und alle Aufrufe zu `input()` und `raw_input()` durch die Debug-Server-Maschinerie umgeleitet. Dieser Code macht zwei Dinge: (1) Alle Warteaufrufe für `sys.stdin` werden im Multiplex-Betrieb ausgeführt, wobei `sys.stdin` und das Debug-Netzwerk-Socket gleichzeitig bedient werden, so dass der Debug-Prozess zu Wing IDE ansprechbar bleibt, während auf Tastatureingaben gewartet wird, und (2) in manchen Fällen wird I/O zu einem anderen Fenster umgeleitet.

Für Debug-Prozesse, die von Wing aus gestartet werden, erscheint der I/O der Tastatur immer im Werkzeug Debug-I/O oder in einer neuen, externen Konsole, die vor dem Starten des Debug-Prozesses erstellt wird. In **Externe I/O-Konsolen** ist beschrieben, wie dies gesteuert werden kann.

Debug-Prozesse, die außerhalb von Wing unter Verwendung von `wingdbstub` gestartet werden, nehmen Ihren Tastatur-I/O immer durch die Umgebung, von der Sie gestartet wurden, vor (dies kann ein Konsolen-Fenster, ein Web-Server oder jede andere I/O-Umgebung sein).

Wenn Befehle in den **Debug-Test** eingegeben werden, wird der I/O während der Zeit, in welcher der Befehl verarbeitet wird, vorübergehend zum Debug-Test umgeleitet.

### 6.13.1. Externe I/O-Konsolen

Wing IDE sammelt die Ausgabe von Ihrem Debug-Prozess standardmäßig im Werkzeug Debug-I/O. Dies ist auch die Stelle, an der Sie Tastatureingaben vornehmen können, wenn Ihr Debug-Programm irgendwelche anfordert.

In Fällen, in denen der Debug-Prozess spezielle Merkmale, die von der Windows-Konsole oder der spezifischen Linux/Unix-Shell bereitgestellt werden, erfordert, können Sie stattdessen den I/O des Debuggens in ein neues, externes Fenster umleiten, wenn Sie die Einstellung **Externe Konsole verwenden** verwenden.

Die wirkungsvollste Art und Weise, dass die externe Konsole nach dem Beenden des Debug-Prozesses sichtbar bleibt, besteht darin, einen Haltepunkt an der letzten Zeile Ihres Programms zu platzieren. Alternativ können Sie die Option **Externe Konsole wartet auf Beenden** auf Wahr setzen. Dies kann jedoch dazu führen, dass sofort viele externe Konsolen angezeigt werden, wenn Sie nicht innerhalb der Konsolen nach jedem Debug-Durchlauf die Eingabetaste drücken.

In Linux/Unix können Sie auswählen, welche Konsole-Anwendungen für die externe Konsole probiert werden, indem Sie die Einstellung **Externe Konsolen** ändern.

Windows verwendet immer die Standard-DOS-Konsole, die mit Ihrer Windows-Version kommt.

### 6.13.2. Multiplex-Betrieb des Debug-Prozess-I/Os deaktivieren

Wenn Sie nur I/O-Aufrufe auf Python-Ebene in Ihrem Programm verwenden, müssen Sie nicht wissen, wie Wing die I/O-Umgebung Ihres Debug-Programms verändert, weil es die Umgebung, die außerhalb des Debuggers gefunden wird, imitiert. Es gibt jedoch mehrere Fälle, welche Benutzer beeinflussen können, die I/O auf Python-Ebene umgehen, indem Sie I/O auf C/C++ Ebene innerhalb eines Erweiterungsmoduls vornehmen:

- Jeder beliebige C/C++ Erweiterungsmodul-Code, der Standard-I/O-Aufrufe unter Verwendung der C-Level `stdin` oder `stdout` vornimmt, wird Wing's I/O-Umgebung umgehen (was nur Python-Level `stdin` und `stdout` beeinflusst). Das bedeutet, dass das Warten auf `stdin` in C oder C++ Code bewirkt, dass der Debug-Prozess auf Wing nicht reagiert und es führt zum Abschalten und Beenden der Debug-Sitzung, wenn Sie versuchen, zu dieser Zeit anzuhalten oder Haltepunkte zu ändern. In diesem Fall wird auch die Umleitung von I/O zum I/O-Werkzeug des Debuggers und dem Debug-Test nicht funktionieren.
- Auf allen Plattformen kann der Aufruf von C-Level `stdin` von mehrfachen Threads in einem multi-threaded Programm zu geänderter Zeichenlesefolge führen, wenn unter dem Wing Debugger ausgeführt wird.
- Wenn in win32 gedebuggt wird, kann das Aufrufen der C-Level `stdin` selbst in einem single-threaded Programm zu einer Race Condition mit Wing's I/O-Multiplexer führen, die zu einer veränderten Lesereihenfolge der Zeichen führt. Dies ist ein unvermeidbares Ergebnis der Beschränkungen vom Multiplexen von Tastatur- und Socket-I/O auf dieser Plattform.

Wenn mit dem Tastatur-I/O Probleme auftreten, sollten Sie dies tun:

1) Schalten Sie Wing's I/O-Multiplexer ab, indem Sie die Einstellung **sys.stdin Wrapper verwenden** auf `Falsch` setzen.
2) Schalten Sie die Option **Externe Konsole verwenden** an (für Einzelheiten siehe **Externe I/O-Konsolen**).

Sobald dies getan ist, sollte I/O in der externen Konsole richtig funktionieren, aber der Debug-Prozess wird weiterhin auf Anhalten- oder Haltepunktbefehle von Wing IDE nicht reagieren, und zwar immer, wenn er auf Eingabe, entweder auf der C/C++ oder der Python-Ebene, wartet.

In diesem Fall wird auch die Tastatureingabe, die als Nebeneffekt von der Verwendung des Debug-Tests aufgerufen wird, durch ungeänderten `stdin` anstatt innerhalb des Debug-Testes passieren, selbst wenn die Befehlsausgabe noch dort erscheint.

## 6.14. Anhängen und Abtrennen

Debug-Prozesse kontaktieren Wing IDE normalerweise automatisch beim Start. Wing IDE kann sich jedoch auch an Debug-Prozesse anhängen, die bereits laufen, aber mit dem IDE noch keinen Kontakt aufgenommen haben, vorausgesetzt der Prozess lässt dies zu. Es gibt zwei Fälle, in denen dies nützlich ist:

- **(1)** Wenn ein extern gestarteter Prozess (einer, der `wingdbstub.py` verwendet, wie im Abschnitt **Extern gestarteten Code debuggen** beschrieben) das IDE an dem konfigurierten Host und Port während dem Anfangsstart nicht erreichen kann, zum Beispiel weil das IDE noch nicht läuft oder nicht konfiguriert wurde, Debug-Verbindungen zu akzeptieren.

- **(2)** Wenn ein Prozess, der an das IDE angehängt ist, mit der Option `Vom Prozess abtrennen` aus dem Menü Debuggen oder mit dem Symbol Abtrennen aus der Werkzeugleiste ausgeschalten wird.

In jedem Fall kann das IDE jede beliebige Anzahl von abgetrennten Prozessen verwalten und wird Ihnen erlauben, es jeweils an einen beliebigen Prozess anzuhängen.

### 6.14.1. Zugriffskontrolle

Wing wird die Funktionalität Anhängen/Abtrennen nur zulassen, wenn es dafür ein Passwort verfügbar hat, das verwendet werden kann, um Zugriffe zu kontrollieren. Das ist sehr wichtig, weil ein ungesicherter Debug-Server dem Client (Wing IDE) volle Kontrolle der Host-Maschine über das Debug-Test-Werkzeug bereitstellt. Jeder beliebige Python-Befehl kann auf diese Weise ausgeführt werden, einschließlich Programmen, welche die Sicherheit Ihrer Maschine und Ihres Netzwerks gefährden.

Da Wing während der Installation ein Zugangspasswort einrichtet, wird Anhängen und Abtrennen „out-of-the-box“ funktionieren, vorausgesetzt, dass Ihre Debug-Prozesse auf eine der folgenden Weisen gestartet werden: Entweder von Wing IDE aus, von Ihnen von der Befehlszeile oder im Zusammenhang mit irgendeinem Dienst oder Programm, das unter Ihrem Benutzernamen auf einer Maschine läuft, die Zugang zu Ihrem **Verzeichnis der Benutzereinstellungen** hat.

Wenn Sie beabsichtigen, entfernt (remote) zu debuggen, müssen Sie außerdem die Datei `.wingdebugpw` aus Ihrem **Verzeichnis der Benutzereinstellungen** in das gleiche Verzeichnis wie `wingdbstub.py` kopieren.

### 6.14.2. Abtrennen

Der Menüpunkt `Vom Prozess abtrennen` im Menü Debuggen wird verwendet, um von einem aktiven Debug-Prozess abzutrennen.

Immer wenn ein Prozess abgetrennt wird, fährt dieser mit der Ausführung fort, so als wenn er außerhalb des Debuggers wäre, ohne an irgendwelchen Haltepunkten oder Exceptions zu stoppen. Selbst wenn ein Prozess zur Zeit des Abtrennens vom IDE innerhalb des Debuggers angehalten ist, wird der Prozess sofort mit der aktiven Ausführung beginnen, nachdem das IDE die Verbindung trennt.

### 6.14.3. Anhängen

Der Menüpunkt `An Prozess anhängen` im Menü Debuggen ist immer dann verfügbar, wenn kein anderer Debug-Prozess an das IDE angehängt ist. Dies schlägt eine Dialogbox auf, die eine Liste mit verfügbaren Prozessen, an die angehängt werden kann, beinhaltet. Diese Liste wird von festverdrahteten Host/Port-Paaren, die mit der Einstellung **Standards anhängen** gegeben sind, kombiniert mit bekannten Prozessen, die vorher an Wing IDE angehängt waren, erstellt.

Wing aktualisiert die Liste der verfügbaren Prozesse, wenn Debug-Sitzungen vom IDE beendet werden, wenn gesehen wird, dass sie von außerhalb beendet werden, während sie an Wing angehängt sind oder wenn der Prozess von Wing nicht kontaktiert werden kann.

Um an einen Prozess anzuhängen, wählen Sie diesen aus der Liste und drücken die Schaltfläche Anhängen. Sie können auch einen Host/Port-Wert manuell eintippen, wenn Ihre Auswahl nicht in der Liste ist (Siehe **Fremde Prozesse identifizieren**).

Sobald Sie an einen Prozess angehängt sind, wird er mit der Ausführung fortfahren, bis ein Haltpunkt oder eine unbehandelte Exception erreicht wird oder bis Sie ihn `Anhalten`.

### 6.14.4. Fremde Prozesse identifizieren

Wenn Sie extern gestarteten Code debuggen (wie in **Extern gestarteten Code debuggen** beschrieben), können Sie die `kAttachPort` Konstante in `wingdbstub.py` verwenden, um den Port, an dem der Debug-Prozess auf Anfragen zum Anhängen von Wing IDE

hören wird, einzustellen. Das ist hilfreich, wenn Sie mehrere Prozesse gleichzeitig ausführen oder in Fällen, in denen der Debug-Prozess nicht in der Lage ist, sich beim Starten an Wing IDE anzuhängen.

Es ist wichtig, für jeden simultanen, extern gestarteten Prozess eindeutige Werte für die `kAttachPort` Konstante zu bestimmen. Wenn der eingestellte Port in Gebrauch ist, wird stattdessen eine zufällige Port-Nummer verwendet und es kann sehr schwierig sein, diese Nummer zu bestimmen, wenn der Prozess Wing IDE am Anfang nicht kontaktieren kann, um sich selbst zu registrieren.

Sobald dies erledigt ist, kann der Debug-Prozess von Wing IDE erreicht werden, indem sein Host/Port in die Textbereiche des Anhängen-Dialogs eingetippt wird. Wenn Sie einen Host/Port-Wert sehr oft eingeben, ist es das Beste, wenn Sie diesen Wert zu der Einstellung **Standards für das Anhängen** hinzufügen.

Siehe Abschnitt **Extern gestarteten Code debuggen** für mehr Informationen.

### 6.14.5. Beschränkungen

Wing unterstützt nur das Anhängen an jeweils einen einzelnen Debug-Prozess. Immer wenn Sie von einem Prozess abtrennen, beginnt er frei zu laufen und wird nicht an Haltepunkten oder nicht-schweren Exceptions stoppen. Dies beschränkt, was mit den Optionen Abtrennen/Anhängen von einer einzelnen Wing-Kopie gemacht werden kann. Wenn Sie zwei Debug-Prozesse zur gleichen Zeit aktiv debuggen möchten und gleichzeitig Schreiten, Haltepunkt-Aktivierung und -Ausführung (wie in einem Client/Server-Netzwerkprogramm) kontrollieren möchten, müssen Sie zwei Kopien von Wing gleichzeitig ausführen.

## 6.15. Extern gestarteten Code debuggen

Dieser Abschnitt beschreibt, wie Sie das Debuggen von einem Prozess, der nicht von Wing gestartet ist, beginnen. Beispiele von Debug-Code, der extern gestartet wird, beinhalten CGI-Skripte oder Web-Servlets, die unter einem Web-Server, Zope oder Plone laufen, und andere eingebettete Python-Skripte, die innerhalb einer größeren Anwendung laufen.

### 6.15.1. Import des Debuggers

Die folgenden, schrittweisen Anweisungen können verwendet werden, um das Debuggen in extern gestartetem Code, der auf der gleichen Maschine wie Wing IDE läuft, zu starten:

1) Kopieren Sie `wingdbstub.py` aus dem Wing IDE Installationsverzeichnis in das gleiche Verzeichnis wie Ihr Debug-Programm.

2) In einigen Fällen müssen Sie auch die Datei `.wingdebugpw` aus Ihrem **Verzeichnis der Benutzereinstellungen** in das gleiche Verzeichnis wie `wingdbstub.py` kopieren. Das ist erforderlich, wenn Sie den Debug-Prozess unter einem anderen Benutzernamen ausführen oder wenn Sie in einer Weise ausführen, die den Debug-Prozess daran hindert, die `.wingdebugpw` Datei innerhalb Ihres `profiles`-Verzeichnisses zu lesen.

3) Fügen Sie an der Stelle, an der Sie mit dem Debuggen beginnen möchten, den folgenden Source-Code ein:

   ```
   import wingdbstub
   ```

   In Abhängigkeit von Ihrer Code-Basis sollten Sie darauf achten, ob diese Anweisung von mehrfachen Prozessen oder Threads erreicht wird. Wenn dies passiert, wird die erste Instanz zu Wing verbinden und die zweite wird beim Verbinden scheitern und ohne Debuggen weiter ausführen.

4) Versichern Sie sich, dass die Wing IDE Einstellung **Passives Hören aktivieren** auf `Wahr` gesetzt ist, um Verbindungen von externen Prozessen zu erlauben.

5) Setzen Sie alle erforderlichen Haltepunkte in Ihrem Python Source-Code.

6) Starten Sie das Debug-Programm außerhalb von Wing IDE, zum Beispiel mit dem Laden einer Seite in Ihrem Web-Browser, wenn das Programm ein CGI-Skript ist. Sie sollten sehen, dass sich die Statusanzeige in den Werkzeugen Stack-Daten, Beobachten und Debug-Test von rot auf gelb oder grün ändert, wie in **Debugger-Status** beschrieben.

   Versichern Sie sich, dass Sie den Python-Interpreter ohne die `-O` Option ausführen. Der Debugger wird nicht funktionieren, wenn die Optimierung angeschalten ist.

7) Der Debugger sollte am ersten Haltepunkt oder an der ersten Exception, die gefunden wird, stoppen. Wird kein Haltepunkt oder keine Exception erreicht, dann wird das Programm bis zum Ende ausführen oder Sie können den Befehl Anhalten aus dem Menü Debuggen verwenden.

**Prozessbeendigung aktivieren**
In einigen Fällen werden Sie die Beendigung von Debug-Prozessen, die außerhalb von Wing IDE gestartet wurden, aktivieren möchten. Wing erkennt standardmäßig extern gestartete Prozesse und deaktiviert die Prozessbeendigung in dieses Fällen, es sei denn, die Einstellung **Extern Gestartete löschen** ist auf `Wahr` gesetzt.

Wenn Sie Probleme haben, dies zum Laufen zu bringen, versuchen Sie `kLogFile` in `wingdbstub.py` variabel zu setzen, um zusätzliche Diagnoseinformationen zu protokollieren.

### 6.15.2. Konfiguration des Debug-Servers

In einigen Fällen müssen Sie auch andere voreingestellte Konfigurationswerte beim Start von `wingdbstub.py` ändern. Diese Werte ersetzen vollständig alle Werte, die in Wing's Projekt- oder Dateieigenschaften eingestellt sind. Diese sind ohnehin nur relevant, wenn das Debug-Programm innerhalb von Wing IDE gestartet wird. Die folgenden Optionen stehen zur Verfügung:

- Der Debugger kann vollständig mit `kWingDebugDisabled=1` ausgeschalten werden. Das ist gleichwertig dem Setzen der `WINGDB_DISABLED` Umgebungsvariable vor dem Starten des Debug-Programms.
- Setzen Sie `kWingHostPort`, um den Netzwerkort von Wing IDE zu bestimmen, so dass der Debugger zu ihm verbinden kann, wenn er startet. Dies ist gleichwertig dem Setzen der `WINGDB_HOSTPORT` Umgebungsvariable vor dem Starten des Debug-Programms. Der voreingestellte Wert ist `localhost:50005`. Siehe Abschnitt **Remote-Debuggen** für Einzelheiten, wenn Sie diesen Wert ändern müssen.
- Mit dem Einstellen von `kLogFile` können Sie steuern, ob interne Fehlernachrichten des Debuggers in eine Protokolldatei geschrieben werden. Verwenden Sie `<stdout>`, `<stderr>` oder einen Dateinamen. Wenn die gegebene Datei nicht existiert, wird sie, wenn möglich, erstellt. Beachten Sie, dass die Verwendung von `<stderr>` in Windows Probleme verursachen kann, wenn der Debug-Prozess nicht in einer Konsole läuft. Dies ist gleichwertig dem Setzen der Umgebungsvariable `WINGDB_LOGFILE` vor dem Starten des Debug-Programms (verwenden Sie einen Wert von `-`, um das Protokollieren zu einer Datei auszuschalten).
- Setzen Sie `kEmbedded` auf `1`, wenn Sie eingebettete Skripte debuggen. In diesem Fall wird die Debug-Verbindung über Skript-Anforderungen aufrechterhalten, anstatt sie zu schließen, wenn das Skript beendet. Wenn dies auf `1` gesetzt ist, müssen

Sie `wingdbstub.debugger.ProgramQuit()` aufrufen, bevor Ihr Programm beendet, um die Debug-Verbindung zum IDE sauber zu schließen. Dies ist gleichwertig dem Setzen der Umgebungsvariable `WINGDB_EMBEDDED`.

- Setzen Sie `kAttachPort`, um den Standard-Port, an dem der Debug-Prozess auf Anfragen zum Anhängen hören wird, zu definieren (nicht verfübar in Wing IDE Personal). Dies ist gleichwertig dem Setzen der `WINGDB_ATTACHPORT` Umgebungsvariable vor dem Starten des Debug-Programms. Wenn dieser Wert kleiner als 0 ist, wird der Debug-Prozess niemals auf Anfragen zum Anhängen hören. Wenn er größer als oder gleich 0 ist, wird dieser Wert verwendet, wenn der Debug-Prozess ohne Kontakt zu Wing IDE läuft, was passieren kann, wenn er am Anfang scheitert, zu dem oben definierten Host und Port zu verbinden oder wenn sich das IDE eine Zeit lang vom Prozess abtrennt. Für Wing IDE Professional ist dies detaillierter im Abschnitt **Anhängen und Abtrennen** beschrieben.

- Setzen Sie `kPWFilePath` und `kPWFileName`, um den Suchpfad und den Dateinamen, die zum Finden einer `.wingdebugpw` Datei für den Debugger verwendet werden, zu definieren. Die Umgebungsvariablen `WINGDB_PWFILEPATH` und `WINGDB_PWFILENAME` werden diese Einstellungen außer Kraft setzen. Der Dateipfad sollte eine Python-Liste von Strings sein, wenn in `wingdbstub.py` eingestellt, oder eine Verzeichnisliste, die durch den Pfadseparator (`os.pathsep`) getrennt ist, wenn von den Umgebungsvariablen gesetzt. Der String `$<winguserprofile>` kann verwendet werden, um Wing's **Verzeichnis der Benutzereinstellungen** für den Nutzer, unter dem der Debug-Prozess läuft, zu definieren. Der Dateiname ist normalerweise `.wingdebugpw`, aber kann in Fällen, in denen diese Bezeichnung ungünstig ist, geändert werden.

- Optional: Setzen Sie `WINGHOME`, welches der Ort des Home-Verzeichnisses der Wing IDE Distribution ist. Dies wird während der Installation eingerichtet, aber kann geändert werden müssen, wenn Sie Wing vom Source-Code ausführen oder die Debugger-Binäre von einer anderen Maschine herüberkopiert haben.

Das Einstellen von irgendwelchen der oben genannten Äquivalente der Umgebungsvariablen wird den Wert, der in der `wingdbstub.py` Datei gegeben ist, außer Kraft setzen.

**Verhalten, wenn das Anhängen ans IDE scheitert**
Immer wenn der Debugger Wing IDE nicht kontaktieren kann (zum Beispiel wenn das IDE nicht läuft oder auf einen anderen Port hört), wird das Debug-Programm ohne zu Debuggen ausführen. Dies ist hilfreich, da debug-aktivierte CGI's und andere Programme normal funktionieren sollten, wenn Wing nicht anwesend ist. Sie können den Debug-Prozess jedoch zwingen, in diesem Fall zu beenden, indem Sie das `kExitOnFailure` Kennzeichen in `wingdbstub.py` setzen. Um an Prozesse, die ohne Debuggen gestartet wurden, anzuhängen, siehe **Anhängen** (nur für Wing IDE Professional).

### 6.15.3. Remote-Debuggen

Da es ziemlich kompliziert ist, das Remote-Debuggen zu konfigurieren, empfehlen wir derzeit, die Remote-Anzeige des IDE's über X Windows (Linux/Unix) oder Remote Desktop (Windows) vorzunehmen, anstatt das IDE auf einem vom Debug-Prozess separaten Host einzurichten.

Wenn dies für Sie keine Option ist, können Sie den Debugger auch so einrichten, dass er entfernt über das Netzwerk verbindet. Um dies zu tun, führen Sie die folgenden Schritte aus (siehe auch **Beispiel für Remote-Debuggen**):

1) Als erstes richten Sie Wing IDE so ein, dass es erfolgreich Verbindungen von einem anderen Prozess innerhalb der gleichen Maschine akzeptiert, wie im Abschnitt **Import des Debuggers** beschrieben. Sie können jedes beliebige Python-Skript zum Testen verwenden, solange Sie Werte haben, die funktionieren.

2) Optional: Ändern Sie die Einstellung **Server-Host** auf den Namen oder die IP-Adresse der Netzwerkschnittstelle, an der das IDE auf Debug-Verbindungen hört. Der voreingestellte Server ist `None`, was anzeigt, dass das IDE auf alle gültigen Netzwerkschnittstellen auf dem Host hören sollte.

3) Optional: Ändern Sie die Einstellung **Server-Port** auf den TCP/IP-Port, an dem das IDE auf Debug-Verbindungen hören sollte. Dieser Wert muss geändert werden, wenn mehrere Kopien von Wing IDE auf dem gleichen Host laufen.

4) Stellen Sie die Einstellung **Erlaubte Hosts** ein, um den Host, auf dem der Debug-Prozess laufen wird, einzuschließen. Aus Sicherheitsgründen wird Wing Verbindungen zurückweisen, wenn der Host hier nicht aufgeführt ist.

5) Installieren Sie als nächstes Wing IDE auf der Maschine, auf der Sie Ihr Debug-Programm ausführen möchten. Eine komplette Wing IDE Installation zu erstellen, ist der einfachste Ansatz.

   Eine Alternative besteht darin, nur den Code des Debug-Servers aus Ihrer primären Wing IDE Installation herauszukopieren. Dies umfasst alle folgenden Dateien und Verzeichnisse unter `WINGHOME`:

   ```
   bin/wingdb.py
   bin/#.#/src/debug/server
   bin/#.#/src.zip/debug/server (nur Python >= 2.3)
   bin/#.#/opensource/schannel (nur Python < 2.3)
   bin/#.#/opensource.zip/schannel (nur Python >= 2.3)
   ```

   Ersetzen Sie #.# mit 1.5, 2.0, 2.1, 2.2, 2.3, und 2.4 (eins für jede unterstützte Python-Version). Wenn Sie nur eine Python-Version verwenden, können Sie die Verzeichnisse für die Versionen, die Sie nicht verwenden, weglassen.

   Die Verzeichnisse innerhalb der Zip-Dateien (nur in Python 2.3 oder höher verwendet) können kopiert werden, indem Sie entweder die gesamte Zip-Datei verschieben oder ein Teilset erstellen, das nur die notwendigen Verzeichnisse enthält.

   Versichern Sie sich, dass Sie diese Verzeichnisse von einer Wing Installation auf den gleichen Host-Typ kopieren, so dass Sie in Linux/Unix `*.so` Erweiterungsmodule, in Windows `*.pyd` Erweiterungsmodule und so weiter einschließen.

6) Als nächstes übertragen Sie Kopien von all Ihrem Debug-Code, so dass die Source-Dateien auf dem Host, auf dem Wing IDE laufen wird, verfügbar sind und dass wenigstens die `*.pyc` Dateien auf dem Debug-Host verfügbar sind.

   Während dem Debuggen müssen die Client- und Server-Kopien Ihrer Source-Dateien übereinstimmen. Ansonsten wird der Debugger scheitern, an Haltepunkten zu stoppen, oder wird am falschen Ort anhalten und das Schreiten durch den Code wird wahrscheinlich nicht richtig funktionieren.

   Da es in Wing IDE keinen Mechanismus zur Übertragung Ihres Codes gibt, müssen Sie NFS, Samba, FTP oder einen anderen Dateitransfer-Mechanismus verwenden, um die entfernten (remote) Dateien aktuell zu halten, wenn Sie sie in Wing bearbeiten.

   Wenn Dateien auf zwei Maschinen an unterschiedlichen Laufwerkorten erscheinen, müssen Sie eine Positionsabbildung der Datei einrichten, was in **Abbildung der Dateiposition** beschrieben ist.

7) Kopieren Sie auf Ihrem Debug-Host `wingdbstub.py` in das gleiche Verzeichnis wie Ihre Source-Dateien und importieren Sie es in Ihren Python-Source-Code, wie in **Extern gestarteten Code debuggen** beschrieben.

8) Wenn Sie `wingdbstub.py` nicht aus einer vollständigen Wing IDE Installation auf den Debug-Host herauskopiert haben, müssen Sie `kWingHome` so einstellen, dass es mit dem Ort, an dem Sie den Debug-Server-Code auf Ihren Debug-Host kopiert haben, übereinstimmt.

9) Setzen Sie `kWingHostPort` in `wingdbstub.py` auf Ihrem Debug-Host. Der Host in diesem Wert muss die IP-Adresse der Maschine, auf der Wing IDE läuft, sein. Der Port muss mit dem Port, der mit der Einstellung **Server-Port** auf dem Host, auf dem Wing IDE läuft, konfiguriert wurde, übereinstimmen.

10) Starten Sie dann Wing neu und versuchen Sie, Ihr Programm auf dem Debug-Host auszuführen. Sie sollten sehen, dass sich die Statusanzeige von Wing's Debugger ändert, um anzuzeigen, dass ein Debug-Prozess angehängt wurde.

Wenn Sie Probleme haben, dies zum Laufen zu bringen, versuchen Sie, `kLogFile` in `wingdbstub.py` variabel zu setzen, um zusätzliche Diagnoseinformationen zu protokollieren.

### 6.15.4. Abbildung der Dateiposition

In Fällen, in denen der vollständige Pfad zu Ihrer Source auf beiden Maschinen nicht der gleiche ist, müssen Sie auch eine Abbildung einrichten, die Wing mitteilt, wo es Ihre Source-Dateien auf jeder Maschine finden kann.

Dies wird mit der Einstellung **Abbildung der Dateiposition** vorgenommen, die die entsprechenden lokalen und Remote-Verzeichnispositionen für die IP-Adresse eines jeden Remote-Hosts, welche in Dezimalnotation angegeben ist (Dotted-Quad-Format), auflistet.

Eine der Host-IP-Adressen innerhalb dieser Einstellung kann auf `"*"` gesetzt werden, um eine Standardabbildung für alle Hosts, die ansonsten in der Abbildung der Dateiposition nicht bestimmt werden, zu definieren.

Jede Host-IP-Adresse in der Abbildung der Dateiposition ist mit einem oder mehreren `(remote_prefix, local_prefix)` Tuples gepaart. Der Remote-Dateivorsatz ist ein vollständiger Pfad auf dem Dateisystem des Debug-Servers. Der lokale Dateivorsatz sollte eine URL sein, die wahlweise mit `file:` beginnt (diese URL sollte keine Backslashes (\) enthalten, selbst wenn der lokale Host eine Windows-Maschine ist) oder er sollte ein Pfadname im UNC-Stil `\\server\share\dir` sein.

Der beste Weg, dies zu verstehen, ist es, einen Blick auf die **Beispiele für die Abbildung der Dateiposition** zu werfen.

Wenn Sie Wing IDE in Windows XP ausführen, können UNC-formatierte Dateinamen, wie `\\machine\path\to\file` verwendet werden. Auf anderen Windows Systemen müssen Sie Remote-Laufwerke zu einem Laufwerkbuchstaben, wie `F:`, abbilden. In Fällen, in denen die Einrichtung einer ständigen Laufwerkabbildung ein Problem darstellt, können Sie ein `cmd.exe` Skript mit einem `net use` Befehl verwenden, um das Laufwerk bei Bedarf abzubilden.

Beachten Sie, dass das Erstellen symbolischer Links auf dem Client oder Server nicht als eine Alternative zur Verwendung dieser Abbildung funktionieren wird. Dies ist ein Nebeneffekt der Funktionalität im Debugger, die sicherstellt, dass das Debuggen richtig funktioniert, wenn symbolische Links vorhanden sind: Intern werden Source-Dateinamen immer zu ihrer genauen, vollständigen Pfadposition aufgelöst.

### 6.15.4.1. Beispiele für die Abbildung der Dateiposition

Der beste Weg, die Positionsabbildung zu verstehen, ist es, ein paar Beispiele zu analysieren.

**Erklärung der Voreinstellungen**

Der voreingestellte Wert der Einstellung **Abbildung der Dateiposition** enthält einen Eintrag für `127.0.0.1`, bei dem die Abbildung auf `None` gesetzt ist (in Python ist dies als `{'127.0.0.1':None}` dargestellt). Dies ist gleichbedeutend mit der ausführlicheren Python-Darstellung von `{'127.0.0.1':[('/','file:')]}`. Es konvertiert vollständige Pfade auf dem Debug-Server in die Client-seitigen URLs, ohne irgendeinen Teil des vollständigen Pfades zu verändern.

**Zwei Linux/Unix-Hosts**

Hier ist eine Beispieldarstellung für `debug.location-map`, die verwendet werden würde, wenn Wing auf `desktop1` ausgeführt wird und einiger Code auf `server1` mit der IP-Adresse `192.168.1.1` gedebuggt wird:

```
debug.location-map={
  '127.0.0.1':None,
  '192.168.1.1':[('/home/apache/cgi', 'fi-
le:/svr1/home/apache/cgi')]
}
```

In diesem Beispiel sind die Dateien, die in `/home/apache/cgi` auf `server1` gelegen sind, die gleichen wie die, die in `/server1/home/apache/cgi` auf `desktop1` zu sehen sind, da das gesamte Dateisystem auf `server1` via NFS gemeinsam genutzt wird und auf `desktop1` unter `/svr1` angebracht wird.

Um diesen Wert in den Einstellungen im GUI einzugeben, würden Sie `192.168.1.1` als eine neue Remote IP-Adresse und ein einzelnes Lokal/Remote-Abbildungspaar, das `/home/apache/cgi` und `file:/svr1/home/apache/cgi` enthält, hinzufügen.

**IDE auf Linux/Unix mit dem Debug-Prozess auf Windows**

Wenn Sie zwischen Windows und Linux oder Unix debuggen, ist bei der Bestimmung der Konvertierungspfade etwas Sorgfalt erforderlich, da auf jeder Plattform unterschiedliche Konventionen bezüglich der Pfadnamen herrschen. Sie würden den folgenden Eintrag verwenden, wenn Wing IDE auf einem Linux/Unix-Host und der Debug-Prozess auf einem Windows-Host mit der IP-Adresse `192.168.1.1` läuft:

```
debug.location-map={
  '127.0.0.1':None,
  '192.168.1.1':[(r'e:\src', 'file:/home/myuser/src')],
}
```

In diesem Beispiel wird das Linux/Unix-Verzeichnis `/home/myuser` mittels Samba mit der Windows-Maschine geteilt und auf dem `e:` Laufwerk abgebildet.

In den Einstellungen im GUI würden Sie `192.168.1.1` als eine neue Remote IP-Adresse und ein einzelnes Lokal/Remote-Abbildungspaar, das `e:\src` und `file:/home/myuser/src` enthält, hinzufügen.

**IDE auf Windows mit dem Debug-Prozess auf Linux/Unix**

Wenn Sie Wing IDE auf einem Windows-Host und den Debug-Prozess auf einem Linux/Unix-Host mit der IP-Adresse `192.168.1.1` ausführen, dann wird statt der gleichen Dateipositionen folgendes verwendet:

```
debug.location-map={
  '127.0.0.1':None,
  '192.168.1.1':[('/home/myuser/src', 'file:e:/src')],
}
```

Nochmals: Beachten Sie die Verwendung von Forwardslashes in der URL, obwohl die Datei auf einer Windows-Maschine ist.

In den Einstellungen im GUI würden Sie `192.168.1.1` als eine neue Remote IP-Adresse und ein einzelnes Lokal/Remote-Abbildungspaar, das `/home/myuser/src` und `file:/e:/src` enthält, hinzufügen.

**Zwei Windows-Hosts**

Wenn Sie Wing IDE auf Windows und den Debug-Prozess auf einer anderen Windwos-Maschine mit der IP-Adresse `192.168.1.1` ausführen, würde folgendes verwendet werden:

```
debug.location-map={
  '127.0.0.1':None,
  '192.168.1.1':[(r'c:\src', 'file:e:/src')],
}
```

In diesem Fall hat der Host, auf dem Wing ausgeführt wird, das komplette `c:` Laufwerk des Remote-Hosts (Debug-Prozess) auf `e:` abgebildet.

In den Einstellungen im GUI würden Sie `192.168.1.1` als eine neue Remote IP-Adresse und ein einzelnes Lokal/Remote-Abbildungspaar, das `c:\src` und `file:e:/src` enthält, hinzufügen.

**Zwei Windows-Hosts, die einen UNC-Share verwenden**

Ein Pfadname im UNC-Stil kann auf Windows XP folgendermaßen verwendet werden:

```
debug.location-map={
  '127.0.0.1':None,
  '192.168.1.1':[(r'c:\src', '\\server\share\dir')],
}
```

In diesem Fall kann auf `c:\src` auf dem Remote-Host, auf dem der Debug-Prozess ausgeführt wird, von der Maschine, auf der Wing IDE ausgeführt wird, über `\\server\share\dir` zugegriffen werden.

In den Einstellungen im GUI würden Sie `192.168.1.1` als eine neue Remote IP-Adresse und ein einzelnes Lokal/Remote-Abbildungspaar, das `c:\src` und `\\server\share\dir` enthält, hinzufügen.

## 6.15.5. Beispiel für das Remote-Debuggen

Hier ist ein einfaches Beispiel, welches das Debuggen als einen Prozess aktiviert, der auf einem Linux/Unix-Host (`192.168.1.200`) läuft und Wing IDE verwendet, das auf einer Windows-Maschine (`192.168.1.210`) läuft. Dieses Beispiel ist nur für wingdbstub-Nutzer. Wenn Sie das WingDBG-Produkt verwenden, um Zope-Code zu debuggen, dann lesen Sie bitte das **Zope Debuggen How-To** (auch in dem Hilfe-Reiter des WingDBG-Produktes enthalten).

Auf der Windows-Maschine müssen die folgenden Einstellungen festgelegt werden:

- **Passives Hören aktivieren** sollte angeklickt werden
- **Server-Host** sollte auf `Alle Oberflächen` gesetzt werden (dies ist die Voreinstellung)
- **Server-Port** sollte auf 50005 gesetzt werden (dies ist die Voreinstellung)
- **Erlaubte Hosts** sollte geändert werden, indem `192.168.1.200` hinzugefügt wird

Auf der Linux/Unix-Maschine ist der folgende Wert in `wingdbstub.py` erforderlich:

```
kWingHostPort='192.168.1.210:50005'
```

Sobald dies eingestellt ist und Wing neu gestartet wurde, sollten Sie in der Lage sein, Code auszuführen, der auf der Linux/Unix-Maschine `wingdbstub` importiert, und Sie sollten sehen, dass sich die Debug-Verbindung auf der Windows-Maschine aufbaut.

Dann müssen Sie zwischen den beiden Maschinen eine gemeinsame Dateinutzung einrichten (zum Beispiel unter Verwendung von Samba) und müssen auf der Windows-Maschine in den Wing IDE Einstellungen eine Abbildung der Dateiposition festlegen.

Wenn sich Ihr Source-Code auf der Linux/Unix-Maschine in `/home/myuser/mysource` befindet und Sie `/home/myuser` in `e:` auf der Windows-Maschine abbilden, dann würden Sie in Verbindung mit den obigen Einstellungen die folgende Abbildung der Dateiposition verwenden:

```
debug.location-
map=('192.168.1.200': [('/home/myuser/mysource', \
                                    'file:e:/mysource')]}
```

Um diese Abbildung der Dateiposition über das Einstellungsmenü im GUI einzugeben, würden Sie `192.168.1.200` als eine neue Remote-Host-IP hinzufügen und ein einzelnes Abbildungspaar mit `/home/myuser/mysource` und `file:e:/mysource` eingeben.

Siehe **Beispiele für die Abbildung der Dateiposition** für weitere Beispiele.

### 6.15.6. Debugger-API

Ein einfacher API kann verwendet werden, um das Debuggen genauer zu steuern, sobald Sie `wingdbstub.py` das erste Mal importiert haben, wie es im Abschnitt **Import des Debuggers** beschrieben ist.

Dies ist in Fällen nützlich, in denen Sie in der Lage sein wollen, das Debuggen mehrere Male während einem Debug-Durchlauf schnell zu starten und zu stoppen, zum Beispiel zur Vermeidung von Debug-Overhead, außer in einem kleinen Unterbereich Ihres Codes. Es kann außerdem in eingebetteten Skripting-Umgebungen hilfreich sein.

Führen Sie die folgenden Schritte aus, um den API zu verwenden:

1) Konfigurieren und importieren Sie `wingdbstub.py`, wie im Abschnitt **Import des Debuggers** beschrieben.

2) Verwenden Sie danach die Instanz-Variable `wingdbstub.debugger`, um einen der folgenden Aufrufe zu machen:

   - **StartDebug(stophere=0, autoquit=1, connect=1)** -- Debuggen starten, optional zum IDE zurückverbinden und/oder danach sofort stoppen. Setzen Sie `autoquit=0`, um zu vermeiden, dass das Debuggen automatisch beendet, wenn das Programmende ermittelt wird (das ist das gleiche wie `kEmbedded` in `wingdbstub.py` einzustellen).
   - **StopDebug()** -- Debuggen komplett stoppen und Verbindung von Wing IDE trennen. Das Debug-Programm setzt die Ausführung im nicht-Debuggen Modus fort und muss neu gestartet werden, um das Debuggen wiederaufzunehmen.
   - **SuspendDebug()** -- Dies lässt die Verbindung zum Debug-Client intakt, aber schaltet den Debugger ab, so dass Verbindungsaufwand während der folgenden Ausführung vermieden wird.
   - **ResumeDebug()** -- Dies wird das Debuggen unter Verwendung einer bestehenden Verbindung zu Wing wiederaufnehmen.
   - **ProgramQuit()** -- Dies muss aufgerufen werden, bevor das Debug-Programm beendet wird, wenn `kEmbedded` in `wingdbstub.py` auf `1` gesetzt wurde oder wenn im vorhergehenden `StartDebug()` API-Aufruf (wenn vorhanden) `autoquit=0` ist. Dies stellt sicher, dass die Debug-Verbindung zum IDE sauber geschlossen wird.

Hier ist ein einfaches Anwendungsbeispiel:

```
import wingdbstub
a = 1 # Diese Zeile wird gedebuggt
wingdbstub.debugger.SuspendDebug()
x = 1 # Diese Zeile wird ohne Debuggen ausgeführt
wingdbstub.debugger.ResumeDebug()
y = 2 # Diese Zeile wird wieder gedebuggt
```

`SuspendDebug()` und `ResumeDebug()` können so oft wie gewünscht aufgerufen werden und verschachtelte Aufrufe werden so verarbeitet, dass das Debuggen nur wiederaufgenommen wird, wenn die Anzahl von `ResumeDebug()` Aufrufen mit der Anzahl der `SuspendDebug()` Aufrufe übereinstimmt.

## 6.16. Ohne Debuggen ausführen

Dateien können auch außerhalb des Debuggers ausgeführt werden. Dies kann mit beliebigem Python-Code, Makefiles und jeder beliebigen anderen Datei, die als auf dem Laufwerk ausführbar gekennzeichnet ist, gemacht werden. Dies kann mit den Optionen `Aktuelle Datei ausführen` und `Letzte ausführen` im Menü `Debuggen` vorgenommen werden oder mit `Ausgewählte ausführen` nach einem rechten Mausklick auf die Projektansicht.

Dateien, die auf diese Weise ausgeführt werden, laufen in einem separaten Prozess und jegliche Eingabe und Ausgabe erscheint in dem Fenster, von dem Wing gestartet wurde (oder ist vollständig versteckt, wenn Wing von einem Desktop-Symbol gestartet wurde).

Dies ist nützlich für das Auslösen von Builds, für die Ausführung von Hilfsprogrammen, die bei der Entwicklung verwendet werden, oder sogar zum Starten eines Programms, das normalerweise außerhalb von Wing gestartet wird und `wingdbstub.py` zum Debuggen verwendet.

Beachten Sie, dass Dateien, die auf diese Weise ausgeführt werden, immer so wie von ihrem aktuellen Verzeichnis und ohne Parameter aufgerufen werden. Es gibt zur Zeit keine Einrichtung zur Bestimmung von Parametern oder für das Umleiten von Eingabe/Ausgabe.

## 6.17. Beschränkungen des Debuggers

Es gibt bestimmte Situationen, die der Debugger nicht verarbeiten kann. Dies liegt an der Art und Weise, wie die Programmiersprache Python funktioniert. Wenn Sie Probleme haben, den Debugger zum Stoppen an Haltepunkten zu bewegen oder Source-Code anzuzeigen, während Sie durch den Code schreiten, dann kann einer oder mehrere dieser Punkte zutreffen.

Lesen Sie immer zuerst den Abschnitt **Fehlerbehebung für Debug-Fehler**. Wenn dies scheitert, Ihr Problem zu bestimmen, dann lesen Sie die folgende detaillierte Dokumentation über die Beschränkungen des Debuggers:

- Ihre Source-Dateien müssen auf dem Laufwerk gespeichert werden und für das IDE

zugänglich sein. Wenn Sie versuchen, Code-Fragmente zu debuggen, versuchen Sie, diese vorübergehend auf dem Laufwerk zu speichern und die `__file__` Variable im Modulnamensbereich zu setzen, bevor Sie Python's `exec` oder `eval` aufrufen.

- Ausführen ohne zu speichern wird zur fehlerhaften Anzeige von Haltepunkten und der Ausführungsposition führen, weil der Debug-Prozess gegen die auf dem Laufwerk vorhandene Version der Source-Datei läuft. Wing wird im Nachrichten-Werkzeug und in der Statusanzeige der Stack-Daten anzeigen, dass einige Dateien keinen `sync` mehr haben; dieser Fall sollte also nur auftreten, wenn Sie die Warnungen ignorieren.

- Sie können das Debug-Programm nicht mit den `-O` oder `-OO` Optimierungsoptionen für den Python-Interpreter ausführen. Dies entfernt Informationen über Zeilennummern und Source-Dateinamen und macht es unmöglich, an Haltepunkten zu stoppen oder durch Code zu schreiten.

- Es gibt verschiedene Fälle, in denen Wing daran scheitert, an Haltepunkten oder Exceptions zu stoppen, oder fehlschlagen kann, mit den Source-Dateien zusammenpassende Haltepunkte oder Exception-Punkte zu finden. Sie werden alle durch die Speicherung von fehlerhaften Dateinamen in `*.pyc` Dateien verursacht:

  – Das Verschieben von `*.pyc` Dateien auf dem Laufwerk nachdem sie erzeugt wurden, annulliert den Dateinamen, der in der Datei gespeichert ist, wenn es ein teilweise relativer Pfad ist. Dies passiert, wenn Ihr `PYTHONPATH` oder `sys.path` teilweise relative Pfadnamen enthält.

  – Ein ähnliches Problem kann aus der Nutzung von `compileall.py` oder einigen anderen Hilfsprogrammen resultieren, die keinen korrekten Dateinamen in der `*.pyc` Datei aufzeichnen.

  – Wenn Sie den gleichen Code zweimal ausführen und dabei unterschiedliche Pfade zum gleichen Arbeitsverzeichnis verwenden, wie es in Linux/Unix mit symbolischen Links möglich ist, können die Dateinamen, die in `*.pyc` belassen werden, eine Mischung von diesen Pfaden enthalten. Wenn der symbolische Link, der verwendet wurde, danach entfernt wird, werden einige der Dateinamen ungültig.

  Die Behebung für alle diese Probleme besteht darin, die `*.pyc` Dateien zu entfernen und sie von Python aus den entsprechenden `*.py` Dateien mit den korrekten Informationen für die Dateinamen neu erstellen zu lassen.

  Tipp: Sie können `*.pyc` Dateien in den meisten Texteditoren öffnen, um gespeicherte Dateinamen zu prüfen.

- In Code, der viel Zeit in C/C++ verbringt, ohne Python überhaupt aufzurufen, zum Beispiel in einer GUI-Hauptschleife, kann es sein, dass der Debugger nicht zuverlässig an Haltepunkten, die während der Ausführung hinzugefügt wurden,

stoppt oder nicht auf Anfragen zum Anhalten antwortet. Siehe Abschnitt **Nicht-Python Hauptschleifen debuggen** für zusätzliche Informationen.

- Sie können `pdb` in Code, den Sie innerhalb des Wing Debuggers ausführen, nicht verwenden. Die zwei Debugger stehen miteinander in Konflikt, weil sie versuchen, die gleichen Debugger-Hooks im Python-Interpreter zu verwenden.
- Wenn Sie `__import__` in Ihrem Code außer Kraft setzen, werden Sie die Fähigkeit des Debuggers, an Haltepunkten zu stoppen, aufheben, es sei denn, Sie rufen das Originale `__import__` als Teil Ihres Codes auf, immer wenn ein Modul tatsächtlich importiert wird. Wenn Sie das Originale `__import__` aus irgendeinem Grund nicht aufrufen können, kann es möglich sein, stattdessen `wingdbstub` zu verwenden, und dann `wingdbstub.debugger.NotifyImport(mod)` von Ihrem Import-Handler aufzurufen (wobei `mod` das Modul ist, das gerade importiert wurde).
- Wenn Sie `__file__` im Namensbereich eines Moduls auf einen Wert anders als den originalen setzen, wird Wing nicht in der Lage sein, an Haltepunkten im Modul zu stoppen und kann scheitern, Exceptions an die Benutzeroberfläche des IDE's zu berichten.
- Wenn Sie ein Erweiterungsmodul verwenden, um `stdio` Aufrufe auf C/C++ Ebene vorzunehmen, anstatt die Einrichtungen auf Python-Ebene zu verwenden, wird der Debug-Prozess weiterhin nicht auf Wing IDE reagieren, während auf Tastatureingabe gewartet wird, die I/O-Umleitung zum Debug-Test wird fehlschlagen und in einigen Fällen kann es zu einer veränderten Lesereihenfolge der Zeichen führen. Einzelheiten sind in **Debug-Prozess I/O** zu finden.
- Die Verwendung von teilweisen Pfadnamen in Modul `__file__` Attributen kann in seltenen Fällen verursachen, dass Wing scheitert, an Haltepunkten und Exceptions zu stoppen, Source-Dateien anzuzeigen oder dass es Source-Dateien mit gleichen Namen durcheinanderbringt.

  Ein teilweiser Pfadname kann nur in `__file__` enden, wenn Sie (a) Python-Code mit einem teilweisen Pfadnamen aufrufen, zum Beispiel mit `python myfile.py` anstelle von `python /path/to/myfile.py`, (b) teilweise Pfadnamen an `exec` senden, (c) teilweise Pfadnamen in Ihrem `PYTHONPATH` oder `sys.path` verwenden oder (d) `compileall.py` oder ähnliche Werkzeuge verwenden, um Module mit teilweisen Pfadnamen zu kompilieren.

  Da Wing alles mögliche unternimmt, um dieses Problem in der Praxis zu vermeiden, tritt es tatsächlich nur in den folgenden seltenen Fällen auf:

  - Wenn Module mit teilweisen Pfadnamen geladen werden und `os.chdir()` aufgerufen wird, bevor das Debuggen gestartet wird. Dies ist nur möglich, wenn `wingdbstub` verwendet wird oder wenn das Debuggen anders gestartet wird, nachdem Ihr Debug-Prozess gestartet wurde.

  - Wenn Module mit teilweisen Pfadnamen geladen werden und `os.chdir()` nach `wingdbstub.debugger.SuspendDebug()` und vor `wingdbstub.debugger.ResumeDebug()` aufgerufen wird.
  - Wenn Module mit teilweisen Pfadnamen geladen werden und von `sys.modules` entfernt werden, bevor der Debugger gestartet wird oder während das Debuggen unterbrochen ist.
  - Wenn Code-Objekte unter Verwendung von `compile()`, dem C-API oder dem neuen Modul schnell erstellt werden, wird ein relativer Dateiname oder ein fehlerhafter Dateiname für das Argument des Dateinamens verwendet und `os.chdir()` wird aufgerufen, bevor der Code ausgeführt wird.

- Wing versucht zu kennzeichnen, wenn Source-Code im IDE mit dem Code, der im Debug-Prozess ausgeführt wird, übereinstimmt oder nicht. Es gibt bestimmte, sehr seltene Fälle, in denen dies scheitert. Dies kann dazu führen, dass das Stoppen an Haltepunken fehlschlägt oder dass andere Probleme auftreten, selbst wenn Dateien vom IDE als synchronisiert gekennzeichnet werden:

  Die Verwendung von `execfile()`, `eval()` oder `exec` mit einem globalen Dictionary, das `__file___` enthält, wird verursachen, dass Wing fehlerhafterweise geltend macht, dass die bestimmte Datei neu geladen wurde. In der Praxis tritt dieses Szenario normalerweise auf, wenn `execfile()` vom Top-Level eines Moduls aufgerufen wird, in welchem Fall das Modul tatsächlich geladen oder neu geladen wird (es tritt also keine Fehlidentifizierung des Ladestatus des Moduls auf). Aber in Fällen, in denen das Laden eines Moduls sehr lange dauert oder eine langlaufende Schleife einbezieht, können `execfile()`, `eval()` oder `exec` auftreten **nachdem** Bearbeitungen am Modul gemacht und gespeichert wurden. In diesem Fall wird Wing das Modul als mit den neuen Bearbeitungen neu geladen fehlidentifizieren.

  Dieses Problem kann auch ausgelöst werden, wenn eine `globals` mit `__file__` ausdrücklich zu `execfile()`, `eval()` oder `exec` gegeben wird. Dies wird in diesem Fall jedoch nur auftreten, wenn der Dateiname des Code-Objektes `?` ist und `locals` und `globals` Dictionaries die gleichen sind, wie sie es standardmäßig für diese Aufrufe sind.

- In sehr seltenen Fällen, wenn Sie `wingdbstub.py` verwenden und `sys.exitfunc` nach dem Starten des Debuggens gesetzt haben, wird das IDE an einer nicht funktionierenden Netzwerkverbindung abschalten, nachdem das Debug-Programm an einer Exception beendet. Dies passiert nur für Exceptions, die aussehen, als ob sie verarbeitet werden, weil ein Try/Except-Block vorhanden ist, der die Exception verarbeiten könnte, aber wobei die Exception am Ende nicht verarbeitet wird und das Debug-Programm am Ende ohne den `StopDebug()` Aufruf beendet. Workarounds beinhalten das Setzen von `sys.exitfunc` bevor `wingdbstub.py` importiert wird oder das Hinzufügen einer Try/Except-Klausel auf dem Top-Level, die vor dem Beenden des Debug-Programms immer `StopDebug()` aufruft.

- Das Benennen einer Datei als `<string>` wird den Debugger vom Debuggen dieser Datei abhalten, weil er mit dem Standard-Dateinamen, der in Python für Code, der nicht in einer Datei platziert ist, verwendet wird, durcheinander kommt.

# Scripting and Extending Wing IDE

Wing IDE provides an API that can be used to extend and enhance the IDE's functionality with scripts written in Python.

> **Important Note**
> Scripting is an experimental feature in this version of Wing IDE. Some portions of the API and other aspects of the scripting facility are subject to change until Wing IDE 2.1. See **Known Scripting Issues** for some details.

Simple scripts can be written without any extra tools -- Wing will find and load scripts at startup and reload them if they change on disk. The API Wing provides allows scripts access to the editor, debugger, project, and a range of application-level functionality. Scripts may also access all **documented preferences** and can issue any number of **documented commands** which implement functionality not duplicated in the formal Python API.

Scripts can be executed like any other command provided by Wing IDE. Scripts can add themselves to the editor and project context menus, or to new menus in the menu bar, and they can also register code for periodic execution as an idle event. They can also be bound to a key combination, or can be invoked by name using the `command-by-name` command.

Errors encountered while loading or executing scripts are displayed in the Scripts channel of the **Messages** tool.

More advanced scripting, including the ability to add tool panels, is also available but generally requires running a copy of Wing IDE from source code, so that scripts can be debugged more efficiently.

## 7.1. Scripting Example

The scripting facility is documented in detail in the sections that follow, but in most cases it is easiest simply to work from the examples in the `scripts` directory in the Wing IDE installation, using the rest of this chapter as a reference.

User scripts are usually placed inside a directory named `scripts` within the **User Settings Directory**. They can also be placed in `scripts` inside the Wing IDE installation.

Try adding a very simple script now by pasting the following into a file called `test.py` within one of the `scripts` directories:

```
import wingapi
def test_script(test_str):
  app = wingapi.gApplication
  v = "Product info is: " + str(app.GetProductInfo())
  v += "\nAnd you typed: %s" % test_str
  wingapi.gApplication.ShowMessageDialog("Test Message", v)
```

Then select Reload All Scripts from the Edit menu. This is only needed the first time a new script file is added, in order to get Wing to discover it. Afterward, Wing automatically reloads scripts whenever they are saved to disk.

Next execute the script by typing `Escape` followed by `X` again and then `test-script`. Wing will ask for the argument `test_str` using it's builtin argument collection facility. Type a string and then `Enter`. The script will pop up a modal message dialog.

Next make a trivial edit to the script (e.g., change „And you typed“ to „Then you typed“). Save the script and execute the script again. You will see that Wing has automatically reloaded the script and the new text appears in the message dialog.

Finally, make an edit to the script that introduces an error into it. For example, change `import wingapi` to `import wingapi2`. Save the script and Wing will show a clickable traceback in the `Scripts` channel of the **Messages** tool. This makes it easy to quickly find and fixed errors in scripts during their development.

That's all there is to basic scripting. The most relevant examples for most simple scripts can be found in `editor_extensions.py` in the `scripts` directory inside the Wing IDE installation. This shows how to access and alter text in the current editor, among other things.

For more advanced scripting, where a more complete debugging support is needed, you will need to obtain a copy of the Wing IDE source code distribution and run Wing from source code so that the scripts (and all of Wing) can be debugged with another copy of Wing (usually your binary installation of Wing).

# 7.2. Getting Started

Scripts are Python modules or packages containing one or more Python functions. When Wing starts up, it will search all directories in the configured **Script Search Path** for modules (`*.py` files) and packages ( directories with an `__init__.py` file and any number of other `*.py` files or sub-packages).

Wing will load scripts defined in each file and add them to the command set that is defined internally. The script directories are traversed in the order they are given in the preference and files are loaded in alphabetical order. When multiple scripts with the same name are found, the script that is loaded last overrides any loaded earlier under that name.

## Naming Scripts

Scripts can be referred to either by their short name or their fully qualified name (FQN).

The short name of a script is the same as the function name but with underscores optionally replaced by dashes (`cmdname.replace('_', '-')`).

The FQN of a script always starts with `.user.`, followed by the module name, followed by the short name.

For example, if a script named `xpext_doit` is defined inside a module named `xpext.py`, then the short name will be `xpext-doit` and the FQN will be `.user.xpext.xpext-doit`.

## Reloading Scripts

Once script files have been loaded, Wing watches the files for changes on disk and automatically reloads them as needed. As a result, there is usually no need to restart Wing when working on a script, except when a new script file is added. In that case, Wing will not load the new script until the `reload-scripts` command (`Reload All Scripts` in the `Edit` menu) is issued or the IDE is restarted.

For details on how reloading works, see **Advanced Scripting**.

## Overriding Internal Commands

Wing will not allow a script to override a command that Wing defines internally (those documented in the **Command Reference**). If a script is named the same as a command

in Wing, it can only be invoked using its fully qualified name. This is a safeguard against completely breaking the IDE by adding a script.

One implication of this behavior is that a script may be broken if a future version of Wing ever adds a command with the same name. This can generally be avoided by using appropriately descriptive and unique names and/or by referencing the command from key bindings and menus using only its fully qualified name.

## 7.3. Script Syntax

Scripts are syntactically valid Python with certain extra annotations and structure that are used by Wing IDE to determine which scripts to load and how to execute them.

Only functions defined at the top level of the Python script are treated as commands, and only those that start with a letter of the alphabet. This allows the use of `_` prefixed names to define utilities that are not themselves commands, and allows use of Python classes defined at the top level of script files in the implementation of script functionality.

### Script Attributes

In most cases additional information about each script `def` is provided via function attributes that define the type of arguments the script expects, whether or not the command is available at any given time, the display name and documentation for the command, and the contexts in which the script should be made available in the GUI.

The following are supported:

- **arginfo** -- This defines the argument types for any arguments passed to the script. It is a dictionary from the argument name to an `ArgInfo` specification (described in more detail below) or a callable object that returns this dictionary. Argument information is used by Wing to drive automatic collection of argument values from the user. When this is missing, all arguments are treated as strings.
- **available** -- This defines whether or not the script is available. If missing, the command is always available. If set to a constant, the truth value of that constant defines availability of the script. If set to a callable object, it is invoked with the same arguments as the script itself and the return value determines availability.
- **label** -- The label to use when referring to the command in menus and elsewhere. When omitted, the label is derived from the command name by replacing underscores with a space and capitalizing each word (`cmdname.replace('_', ' ').title()`)

- **doc** -- The documentation for the script. Usually, a docstring in the function definition is used instead.
- **contexts** -- The contexts in which the script will be added in the GUI, a described in more detail below.

## ArgInfo

Argument information is specified using the `CArgInfo` class in the Wing API (`wingapi.py` inside `bin` in the Wing IDE installation, although the class is imported from Wing IDE's internals) and the `datatype` and `formbuilder` modules in Wing's `wingutils` package. The source code for this class and support modules is only available in the source distribution, although most use cases are covered by the following.

`CArgInfo`'s contructor takes the following arguments:

- **doc** -- The documentation string for the argument
- **type** -- The data type, using one of the classes descended from wingutils.datatype.CTypeDef (see below for the most commonly used ones)
- **formlet** -- The GUI formlet to use to collect the argument from the user when needed. This is one of the classes descended wingutils.formbuilder.CDataGui (see below for the most commonly used ones).
- **label** -- The label to use for the argument when collected from the user. This argument may be omitted, in which case Wing builds the label as for the `label` function attribute described above.

## Commonly Used Types

The following classes in `wingutils.datatype.py` cover most cases needed for scripting:

- **CBoolean** -- A boolean value. Constructor takes no arguments.
- **CType** -- A value of type matching one of the parameters sent to the constructor. For example, `CType("")` for a string, `CType(1)` for an integer, and `CType(1.0, 1)` for a float, or an integer.
- **CValue** -- One of the values passed to the constructor. For example `CValue("one", "two", "three")` to allow a value to be either `"one"`, `"two"`, or `"three"`.

- **CRange** -- A value between the first and second argument passed to the constructor. For example, `CRange(1.0, 10.0)` for a value between `1.0` and `10.0`, inclusive.

Additional types are defined in `wingutils.datatype.py`, but these are not usually needed in describing scripting arguments.

## Commonly Used Formlets

The following classes in `guiutils.formbuilder.py` cover most of the data collection formlets needed for scripting:

- **CSmallTextGui** -- A short text string entry area with optional history, autocompletion, and other options. The constructor takes the following keyword arguments, all of which are optional:

```
maxlen            -- Maximum allowed text length (-
1=any, default=80)
history           --
List of strings for history (most recent 1st) or
                     a callable that will return the histo-
ry (default=None)
choices           --
List of strings with all choices, or a callable
                     that will take a fragment and re-
turn all possible
                     matches (default=None)
partial_complete  --
True to only complete as far as unique match when
                     the tab key is pressed.  Default=True.
stopchars         --
List of chars to always stop partial completion.
                     Default=''
allow_only        --
List of chars allowed for input (all others are
                     not processed).  Set to None to al-
low all.  Default=None
auto_select_choice --
True to automatically select all of the entry text
                     when browsing on the autocomple-
ter (so it gets erased
                     when any typing hap-
pens).  Default=False.
```

```
default           -- The default value to use.  Default=''
select_on_focus   --
True to select range on focus click; false to retain
                     pre-focus selection.  Default=False
editable          -- True to allow editing this field.  Default=True.
```

- **CLargeTextGui** -- A longer text string. The constructor takes no arguments.
- **CBooleanGui** -- A single checkbox for collecting a boolean value. The constructor takes no arguments.
- **CFileSelectorGui** -- A keyboard-driven file selector with auto-completion, optional history, and option to browse using a standard file open dialog. The constructor takes the following keyword arguments:

```
want_dir          --
True to browse for a directory name (instead of a
                     file name).  Default=False.
history           --
Optional list with history of recent choices, most
                     recent first.  Default=()
default           -- The default value to use.  Default=''
```

Additional formlet types are defined in `guiutils.formbuilder.py` but these are not usually needed in collecting scripting arguments.

- **CPopupChoiceGui** -- A popup menu to select from a range of values. The constructor takes a list of items for the popup. Each item may be one of:

```
None                 -- A divider
string               --
The value.  The label used in the menu is derived:
                        label = va-
lue.replace('_', ' ').title()
(value, label)       -- The value and label to use in menu.
(value, label, tip)  --
The value, label, and a tooltip to show when the
                        user hovers over the menu item.
```

- **CNumberGui** -- A small entry area for collecting a number. The constructor takes these arguments (all are required):

```
min_value            -- The minimum value (inclusive)
max_value            -- The maximum value (inclusive)
page_size            --
Increment when scroller is used to browse the range
```

```
num_decimals          -- Number of decimal places (0 to col-
lect an integer)
```

Additional formlets for collecting data are defined in `guiutils.formbuilder.py`, but these are not usually needed for scripting.

## Magic Default Argument Values

Wing treats certain defaults values specially when they are specified for a script's arguments. When these default values are given, Wing will replace them with instances of objects defined in the API. This is a convenient way for the script to access the application, debugger, current project, current editor, and other objects in the API. All the default values are defined in the `wingapi.py` file, as are the classes they reference.

- **kArgApplication** -- The `CAPIApplication` instance (this is a singleton).
- **kArgDebugger** -- The currently active `CAPIDebugger`.
- **kArgProject** -- The currently active `CAPIProject`.
- **kArgEditor** -- The currently active `CAPIEditor`.
- **kArgDocument** -- The `CAPIDocument` for the currently active editor.

## GUI Contexts

Scripts can use the `contexts` function attribute to cause Wing to automatically place the script into certain menus or other parts of the GUI. The following contexts are currently supported (they are defined in `wingapi.py`):

- **kContextEditor** -- Adds an item to the end of the editor's context menu (accessed by right clicking on the editor)
- **kContextProject** -- Adds an item to the end of the project's context menu (accessed by right clicking on the project)
- **kContextNewMenu** -- Adds an item to a new menu in the menu bar. This is a class whose constructor takes the localized name of the menu to add. The menu is only added if one or more valid scripts with that menu context are successfully loaded.

- **kContextScriptsMenu** -- Adds an item to the scripts menu, which is shown in the menu bar if any scripts are added to it (this is currently the same as `kContextNewMenu("Scripts")` but may be moved in the future).

All scripts, under both short and fully qualified name, are always listed along with all internally defined commands in the auto-completion list presented by the `command-by-name` command (bound to keyboard sequence `Esc X` -- `Escape` key followed by `X`) and in the **Custom Key Bindings** preference.

## Top-level Attributes

Default values for some of the Script Attributes defined above can be set at the top level of the script file, and some additional attributes are also supported:

- **_arginfo** -- The default argument information to use when no per-script `arginfo` attribute is present.
- **_available** -- The default availability of scripts when no `available` attribute is present.
- **_contexts** -- The default contexts in which to add scripts when no `contexts` attribute is present.
- **_ignore_scripts** -- When set to True, Wing will completely ignore this script file.
- **_i18n_module** -- The name of the `gettext` internationalized string database to use when translating docstrings in this script. See below for more information.

## Importing Other Modules

Scripts can import other modules from the standard library, `wingapi` (the API), and even from Wing's internals. However, because of the way in which Wing loads scripts, users should avoid importing one script file into another. If this is done, the module loaded at the `import` will not be the same as the one loaded into the scripting manager. This happens because the scripting manager uniquifies the module name by prepending `internal_script_` so two entries in `sys.modules` will result. In practice, this is not always a problem except if global data at the top level of the script module is used as a way to share data between the two script modules. Be sure to completely understand Python's module loading facility before importing one script into another.

### Internationalization and Localization

String literals and docstrings defined in script files can be flagged for translation using the `gettext` system. To do this, the following code should be added before any string literals are used:

```
import gettext
_ = gettext.translation('scripts_example', fallback=1).gettext
_i18n_module = 'scripts_example'
```

The string `'scripts_example'` should be replaced with the name of the `.mo` translation file that will be added to the `resources/locale` localization directories inside the Wing installation.

Subsequently, all translatable strings are passed to the `_()` function as in this code example:

```
kMenuName = _("Test Base")
```

The separate `_i18n_module` attribute is needed to tell Wing how to translate docstrings (which cannot be passed to `_()`).

Currently, the only support provided by Wing for producing the `*.po` and `*.mo` files used in the `gettext` translation system is in the build system that comes with the Wing IDE sources. Please refer to `build-files/wingide.py` and `build-files/README.txt` for details on extracting strings, merging string updates, and compiling the `*.mo` files. On Linux, KDE's `kbabel` is a good tool for managing the translations.

## 7.4. Scripting API

> **Important Note**
> The scripting API is currently experimental and subject to change until Wing IDE version 2.1. See **Known Scripting Issues** for some details.

Wing's formal scripting API consists of several parts:

1) The contents of the `wingapi.py` file in `bin` inside the Wing IDE installation (this file is located in `src` when working from the source distribution). Please refer to the file itself for details of the API.

2) The portions of the `wingutils.datatype` and `guiutils.formbuilder` modules that are documented in the preceding section.

3) All of the **documented commands** which can be invoked using the `ExecuteCommand()` method on `wingapi.gApplication`. Note keyword arguments can be passed to commands that take them, for example `ExecuteCommand('replace-string', search_string="tset", replace_string="test")`

4) All of the **documented preferences** which can be obtained and altered using `GetPreference` and `SetPreference` on `wingapi.gApplication`.

Scripts can, of course, also import and use standard library modules from Python, although Wing ships with a pruned subset of the standard library that includes only those modules that are used by the IDE's internals.

Advanced scripts may also „reach through“ the API into Wing internals, however this requires reading Wing's souce code and no guarantee is made that these will remain unchanged or will change only in a backward compatible manner.

## 7.5. Advanced Scripting

While simple scripts can generally be developed from example using only the Wing IDE binary distribution, more advanced scripts require Wing to be run from the source code distribution, usually as a debug process being controlled by another copy of Wing IDE.

This provides not only more complete access to the source code for scripts that reach through the API into Wing internals, but also more complete support for debugging the scripts as they are developed.

To obtain Wing's source code, you must have a valid license to Wing IDE Professional or higher and must fill out and submit a non-disclosure agreement. Once this is done, you will be provided with access to the source code and more information on working with Wing IDE's sources.

### Example

For an example of an advanced script that adds a tool panel to the IDE's interface, see `templating.py` in the `scripts` directory inside the Wing IDE installation.

## How Script Reloading Works

Advanced scripters working outside of the API defined in `wingapi.py` should note that Wing only clears code objects registered through the API. For example, a script-added timeout (using `CAPIApplication.InstallTimeout()` method) will be removed and re-added automatically during reload, but a tool panel added using Wing internals will need to be removed and re-added before it updates to run on altered script code. In some cases, when object references from a script file are installed into Wing's internals, it will be necessary to restart Wing IDE.

Here is how reloading works:

1) All currently loaded script files are watched so that saving the file from an editor will cause Wing to initiate reload after it has been saved.
2) When a file changes, all scripts in its directory will be reloaded.
3) Wing removes all old scripts from the command registry and registers any timeouts set with `CAPIApplication.InstallTimeout()`.
4) Next `imp.find_module` is used to locate the module by name.
5) Then the module is removed from `sys.modules` and reloaded using `imp.find_module` and a module name that prepends `internal_script_` to the module name (in order to avoid conflicting with other modules loaded by the IDE).
6) If module load fails (for example, due to a syntax error), any timeouts registered by the module during partial load are removed and the module is removed from `sys.modules`.
7) If the module contains `_ignore_scripts`, then its timeouts (if any) are removed and scripts in the file are ignored.
8) Otherwise, Wing adds all the scripts in the module to the command registry and loads any sub-modules if the module is a package with `__init__.py`.

Note that reloading is by design slightly different than Python's builtin `reload()` function: Any old top-level symbols are blown away rather than being retained. This places some limits on what can be done with global data: For example, storing a database connection will require re-establishing the connection each time the script is reloaded.

## 7.6. Known Scripting Issues

Scripting is an experimental feature in this version of Wing IDE and some aspects of the API and the scripting facility in general are likely to undergo some change before scripting becomes an official feature in Wing IDE 2.1. These changes may in some cases require making changes to scripts written with earlier versions of Wing.

If you are scripting Wing IDE, please submit bug reports, feedback, and suggestions from the Help menu. This will help us fix the problems that impact real users, and to design the additional scripting support features that should be added to the IDE in the future.

Although we cannot predict all the changes that may occur, here are some of the known issues that are scheduled for work:

- **URLs are not real URLs** -- The API accepts URLs but there are some limitations and problems with what is accepted: Currently only `file:` URLs will work, and the API does do not accept standard URL encoding (e.g., substituting spaces with `'+'`). Instead, the URLs are essentially the full path to the file with `file:` prepended. Future versions will require real URLs and will add support for some additional protocols.
- **Portions of the API are untested** -- Some parts of the API have not yet been tested because unit tests do not yet exist for it.
- **Cannot add toolbar items** -- There is no support yet for adding toolbar items with scripts.
- **Incomplete GUI integration** -- There is no scripting support tool in the toolset. Messages and errors are shown in the `Scripts` channel of the `Messages` tool, and the Edit menu provides a `Reload All Scripts` item but otherwise scripting is done manually by opening files in the scripting directories (which can of course be added to your project file) and editing them.

# Referenz der Einstellungen

Dieses Kapitel dokumentiert das gesamte Set der verfügbaren Einstellungen für Wing IDE. Die meisten Einstellungen können mit der Option **Einstellungen des GUI** festgelegt werden. Einige Nutzer sind vielleicht daran interessiert, Einstellungsdateien manuell zu erstellen, um verschiedene Instanzen von Wing IDE zu steuern (für Einzelheiten siehe **Anpassung der Einstellungen**).

## Benutzeroberfläche

### System-Gtk verwenden

Die systemweite gtk-Bibliothek verwenden (erfordert gtk 2.2 oder höher). Wing wird mit seiner eigenen Kopie der gtk-Bibliotheken geliefert, für die es erstellt und getestet ist. Verwenden Sie die System-gtk-Option zur besseren Integration mit gnome oder anderen Desktop-Umgebungen. Bei einigen Systemen kann dies jedoch zu zufälligen Abstürzen oder anderen Fehlern führen, die aus Binär-Inkompatibilitäten in Bibliotheksversionen resultieren. Diese Einstellung kann in der Command Line mit dem --system-gtk und --private-gtk Command Line Optionen außer Kraft gesetzt werden.

Interner Name:

```
gui.use-system-gtk
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
False
```

### Anzeigethema

**Konfiguriert den gesamten Anzeigestil oder das Thema, das von Wing IDE verwend**
können unter http://art.gnome.org/themes heruntergeladen werden und in WINGHOME/bin/gtk-bin/share/themes oder USER_SETTINGS_DIR/themes platziert werden. Diese werden unten zu den Auswahlmöglichkeiten hinzugefügt. Es wird jedoch nur die Pixmap Theme Engine unterstützt.

Interner Name:

```
gui.display-theme
```

Daten Spezifikation:

```
[H2O-gtk2-Saphire, Aero, H2O-gtk2-Emerald, H2O-gtk2-
Amber, AluminumAlloy-Toxic, Redmond95, Smooth-2000, H2O-
gtk2-Amythist, HighContrastLargePrint, AluminumAlloy-
Cryogenic, HighContrast, AluminumAlloy-Volcanic, LowCon-
trast, LargePrint, HighContrastLargePrintInverse, AluminumAlloy-
Smog, HighContrastInverse, Smokey-Blue, Glider, Smooth-Sea-
Ice, Default, Glossy P, Redmond, None, Smooth-Retro, Smooth-
Desert, H2O-gtk2-Ruby, LowContrastLargePrint, GnuBubble]
```

Standardeinstellung:

```
None
```

### Anzeigesprache

The language to use for the user interface. Either the default for this system, or set to a specific supported language.

Interner Name:

```
main.display-language
```

Daten Spezifikation:

```
[None, de, en, fr]
```

Standardeinstellung:

```
None
```

**Schriftart/-größe der Anzeige**

Die Basis-Schriftart und -größe, die für die Menüs und Beschriftungen der Benutzeroberfläche verwendet werden.

Interner Name:

```
gui.default-font
```

Daten Spezifikation:

```
[None oder <type str>]
```

Standardeinstellung:

```
None
```

**Schriftart/-größe des Source-Codes**

Die Basis-Schriftart und -größe für die Verwendung im Source-Code-Editor, der Python-Shell, dem Debug-Test, Source-Assistent und anderen Werkzeugen, die Source-Code anzeigen.

Interner Name:

```
edit.default-font
```

Daten Spezifikation:

```
[None oder <type str>]
```

Standardeinstellung:

```
None
```

- **Layout**

### Fensteraufteilung

Grundsätze für die Fenstererstellung: Der Modus des kombinierten Fensters platziert die Werkzeugboxen in Editor-Fenstern; der Modus der separaten Werkzeugbox-Fenster erstellt separate Werkzeugbox-Fenster; der Ein-Fenster-pro-Editor Modus erstellt außerdem ein neues Fenster für jeden Editor.

Interner Name:

```
gui.windowing-policy
```

Daten Spezifikation:

```
[combined-window, one-window-per-editor, separate-toolbox-
window]
```

Standardeinstellung:

```
combined-window
```

### Position der ersten Werkzeugbox

Konfiguriert die Position des hohen Feldbereiches im Hauptanzeigefenster.

Interner Name:

```
gui.tall-panel-location
```

Daten Spezifikation:

```
[right, left]
```

Standardeinstellung:

```
right
```

### Position der zweiten Werkzeugbox

Konfiguriert die Position des breiten Feldbereiches im Hauptanzeigefenster.

Interner Name:

```
gui.wide-panel-location
```

Daten Spezifikation:

```
[top, bottom]
```

Standardeinstellung:

```
bottom
```

**Notizbuchreiter des Editors anzeigen**

Kontrolliert, ob Wing Notizbuch-Reiter für das Wechseln zwischen Editoren anzeigt. Wenn falsch, wird stattdessen ein Popup-Menü verwendet.

Interner Name:

```
gui.use-notebook-editors
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

**Werkzeug-Hinweise aktivieren**

Kontrolliert ob Werkzeug-Tipps, die Hilfe-Informationen beinhalten, angezeigt werden, wenn der Nutzer die Maus über Bereiche der Benutzeroberfläche bewegt.

Interner Name:

```
gui.enable-tooltips
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

- **Werkzeugleisten**

### Werkzeugleiste anzeigen

Ob die Werkzeugleiste in irgendeinem Fenster gezeigt wird.

Interner Name:

```
gui.show-toolbar
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

### Größe der Werkzeugleiste

Stellt die Größe der Werkzeugleistensymbole ein. Entweder „klein“, „mittel“, „groß“ oder„extragroß“ oder verwenden Sie „Standard“, um die systemweiten Einstellungen zu wählen.

Interner Name:

```
gui.toolbar-icon-size
```

Daten Spezifikation:

```
[medium, default, xlarge, text-height, large, small]
```

Standardeinstellung:

```
small
```

### Stil der Werkzeugleiste

Stil der zu verwendenden Werkzeugleistensymbole wählen. Entweder „Nur Symbole“, „Nur Text“, „Symbol un Text nach unten“, „Symbol un Text zur Seite“, oder verwenden Sie „Standard“, um die systemweiten Einstellungen zu wählen.

Interner Name:

```
gui.toolbar-icon-style
```

Daten Spezifikation:

```
[medium, default, xlarge, text-height, large, small]
```

Standardeinstellung:

```
text-right
```

- **Farben**

### Farbe der Textmarkierung

Die Farbe, die verwendet wird, um die aktuelle Textauswahl des zu bearbeitenden Textes anzuzeigen.

Interner Name:

```
gui.text-selection-color
```

Daten Spezifikation:

```
[tuple Länge 3 von: [von 0 bis 255], [von 0 bis 255], [von 0 bis 255]]
```

Standardeinstellung:

```
(253, 253, 104)
```

### Hintergrund des Source-Codes

Hintergrundfarbe für die Verwendung im Source-Editor, der Python-Shell, dem Debug-Test, Source-Assistent und anderen Werkzeugen, die Source-Code anzeigen. Die Vordergrundfarben für den Text werden automatisch geändert, damit sie sich von der Hintergrundfarbe abheben.

Interner Name:

```
edit.background-color
```

Daten Spezifikation:

```
[None oder [tuple Länge 3 von: [von 0 bis 255], [von 0 bis 255], [von 0 bis
```

Standardeinstellung:

```
None
```

**Markierungsfarbe für Debugger-Ausführung**

Die Farbe der Textmarkierung, die für die Ausführungsposition während des Debuggens verwendet wird

Interner Name:

```
debug.run-marker-color
```

Daten Spezifikation:

```
[tuple Länge 3 von: [von 0 bis 255], [von 0 bis 255], [von 0 bis 255]]
```

Standardeinstellung:

```
(255, 163, 163)
```

**Syntax Formatting**

Formatting options for syntax coloring in editors. Colors are relative to a white background and will be transformed if the background color is set to a color other than white.

Interner Name:

```
.edit.syntax-formatting
```

Daten Spezifikation:

```
[dict; keys: <type str>, Werte: [dict; keys: [italic, back, fo-
re, bold], Werte: [einer von: None, <type str>, <boo-
lean: 0 oder 1>]]]
```

Standardeinstellung:

```
{}
```

- **Tastatur**

**Individualität**

Selects editor personality

Interner Name:

```
edit.personality
```

Daten Spezifikation:

```
[vi, visualstudio, emacs, brief, normal]
```

Standardeinstellung:

```
normal
```

**Benutzerdefinierte Tastaturbefehle**

Tastaturbefehle in der Datei für Tastaturbefehle überschreiben. Um einen Tastaturbefehl einzufügen, müssen Sie den Eingabebereich aktivieren und die gewünschte Tastenkombination eingeben. Die Befehle sind im Benutzerhandbuch, Abschnitt Befehlsreferenz, dokumentiert oder sie können den Namen eines benutzerdefinierten Skripts, das in Wing IDE geladen wurde, verwenden.

Interner Name:

```
gui.keymap-override
```

Daten Spezifikation:

```
[dict; keys: <type str>, Werte: <type str>]
```

Standardeinstellung:

```
{}
```

**Zeitabschaltung für Buchstabenfolgen**

Stellt die für die Tastatureingabe verwendete Zeitabschaltung in Sekunden ein, nach der die betätigten Tasten als eine separate Gruppe von Zeichen betrachtet werden. Dies wird für Auswahlmöglichkeiten mit der Tastatur auf Listen oder in anderen GUI Bereichen verwendet. Vor der Zeitabschaltung werden die nachfolgenden Tasten zu den vorherigen hinzugefügt, um die Auswahl während der Tastaturnavigation zu verbessern.

Interner Name:

```
gui.typing-group-timeout
```

Daten Spezifikation:

```
<type float>, <type int>
```

Standardeinstellung:

```
1
```

**VI Mode Ctrl-C/X/V**

Controls the behavior of the Ctrl-X/C/V key bindings in vi mode. Either always use these for cut/copy/paste, use them for vi native actions such as initiate-numeric-repeat and start-select-block, or use the default by system (clipboard on win32 and OS X, and other commands elsewhere).

Interner Name:

```
vi-mode.clipboard-bindings
```

Daten Spezifikation:

```
[other, clipboard, system-default]
```

Standardeinstellung:

```
system-default
```

- **Other**

**Start-Bild anzeigen**

Kontrolliert ob das Einschalt-Bildschirm gezeigt wird

Interner Name:

```
main.show-splash-screen
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

**Case Sensitive Sorting**

Controls whether names are sorted case sensitively (with all caps preceding small letters) or case insensitively

Interner Name:

```
gui.sort-case-sensitive
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

### Fehlerbericht Dialog automatisch zeigen

Bestimmt, ob der Fehlerbericht-Dialog (auch verfügbar im Hilfemenu) dem Benutzer bei unerwarteten Exceptions des IDEs automatisch angezeigt wird.

Interner Name:

```
gui.show-report-error-dialog
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
False
```

### Auto-check for Product Updates

Automatically attempt to connect to wingware.com to check for updates once every day after Wing is started.

Interner Name:

```
main.auto-check-updates
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

- **Erweitert**

### Anzeigebereich

Rechteck, das für den IDE Arbeitsbereich auf dem Bildschirm verwendet wird. Alle Fenster öffnen sich in diesem Bereich. Das Format ist (x, y, Breite, Höhe) oder verwenden Sie None für den Vollbildschirm.

Interner Name:

```
gui.work-area-rect
```

Daten Spezifikation:

```
[None oder [tuple Länge 4 von: <type int>, <type int>, <ty-
pe int>, <type int>]]
```

Standardeinstellung:

```
None
```

**Maximale Größe des Fehlerprotokolls**

Legt Anzahl der Bytes fest, an welcher die Fehlerprotokolldatei (USER_SETTINGS_DIR/error-log) abgeschnitten wird. Diese Datei kann an den technischen Support gesendet werden, um bei der Diagnose von Problemen mit dem IDE zu helfen.

Interner Name:

```
main.max-error-log-size
```

Daten Spezifikation:

```
[von 0 bis 1000000000]
```

Standardeinstellung:

```
100000
```

**Datei der Tastaturbefehle**

Defines location of the keymap override file. Use None for default according to configured editor personality. See the Wing IDE Manual for details on building your keymap override file -- in general this is used only in development or debugging keymaps; use the keymap-override preference instead for better tracking across Wing versions.

Interner Name:

```
gui.keymap
```

Daten Spezifikation:

```
[None oder <type str>]
```

Standardeinstellung:

```
None
```

**Nachrichten**

Kontrolliert das Format und den Umfang von Nachrichten, die dem Nutzer für jede Nachrichtendomäne im Nachrichtenbereich angezeigt werden. Jede Domäne bestimmt das Format (in Python 2.3 logging.Formatter Format) und die minimale Protokollierungsebene, die in der Anzeige gezeigt werden sollte. Wenn eine Nachrichtendomäne nicht spezifiziert wird, dann werden stattdessen die Einstellungen der Parent-Domäne verwendet („“ ist der Parent von allen Domänen).

Interner Name:

```
gui.message-config
```

Daten Spezifikation:

```
[dict; keys: [search, debugger, analysis, general, pro-
ject, editor, scripts, browser], Werte: [tuple Länge 3 von: <ty-
pe str>, [0, 40, 30], <type int>]]
```

Standardeinstellung:

```
{'': ('%(message)s', 0, 100000)}
```

**Stile der Dokumententexte**

Definiert Textstile, die in der Daten- und Dokumentenanzeige verwendet werden. Jeder Stil wird als eine Liste von (Name, Wert) Tuples angegeben. Die Namen und Werte müssen gültige Pango-Textattributnamen und -werte sein. Zur Einstellung von Standardwerten, die auf alle Stile angewendet werden, verwenden Sie den „Standard“ Stilnamen (zum Beispiel ändert das Hinzufügen von („Größe“, 14) die Standard-Anzeigegröße auf 14 Punkte). Beachten Sie, dass die Größe von Menüs, Schältflächen, Labels und anderen grundlegenden GUI-Elementen mit der systemweiten Themenkonfiguration und nicht mit dieser Einstellung gesetzt wird. Der Source-Editor wird auch separat konfiguriert.

Interner Name:

```
main.text-styles
```

Daten Spezifikation:

```
[dict; keys: [einer von: <type str>, [admonition-title, dan-
ger, footnote, citation, admonition, calltip-doc, title-
4, calltip-strong, caution, title-3, title-0, title-
1, image-link, calltip-type, calltip-poc, hint, calltip-arg-
current, tip, literal, note, field, emphasis, title-2, calltip-
class-symbol, attention, calltip-def-symbol, link, strong, marked-
list-items, calltip-def, list-items, default, docinfo-
header, transition, calltip-arg, caption, warning, er-
ror, navigation-link, navigation]], Werte: [tuple von: [ei-
ner von: [tuple Länge 2 von: [foreground], [None oder <ty-
pe str>]], [tuple Länge 2 von: [style], [None oder [ob-
lique, italic, normal]]], [tuple Länge 2 von: [justifi-
cation], [None oder [right, fill, center, left]]], [tu-
ple Länge 2 von: [font_desc], [None oder <type str>]], [tu-
ple Länge 2 von: [weight], [None oder [einer von: <ty-
pe int>, [heavy, bold, ultrabold, normal, light, ul-
tralight]]]], [tuple Länge 2 von: [right_margin], [No-
ne oder [1]]], [tuple Länge 2 von: [stretch], [None oder [con-
densed, expanded, normal, semicondensed, extraconden-
sed, extraexpanded, semiexpanded, ultracondensed, ultra-
expanded]]], [tuple Länge 2 von: [strikethrough], [No-
ne oder <boolean: 0 oder 1>]], [tuple Länge 2 von: [rise], [No-
ne oder [von -100000 bis 100000]]], [tuple Länge 2 von: [va-
riant], [None oder [smallcaps, normal]]], [tuple Län-
ge 2 von: [underline], [None oder [double, single, low, no-
ne]]], [tuple Länge 2 von: [ypad], [None oder [1]]], [tu-
ple Länge 2 von: [background], [None oder <type str>]], [tu-
ple Länge 2 von: [indent], [None oder [1]]], [tuple Län-
ge 2 von: [left_margin], [None oder [1]]], [tuple Län-
ge 2 von: [font_family], [None oder <type str>]], [tuple Län-
ge 2 von: [xpad], [None oder [1]]], [tuple Länge 2 von: [si-
ze], [None oder [einer von: [von 0 bis 1000000], [medium, x-
large, xx-large, large, small, xx-small, x-small]]]]]]]
```

Standardeinstellung:

```
{'calltip-strong': (('font_family', 'sans'), ('weight', 'bold'), ('fo-
reground', '#000066')), 'danger': (('background', '#ffffdd'),), 'foot-
```

```
note’: ((’weight’, ’bold’),), ’navigation-link’: ((’fore-
ground’, ’#909090’), (’style’, ’italic’), (’weight’, ’bold’)), ’ci-
tation’: ((’weight’, ’bold’),), ’admonition’: (), ’list-
items’: ((’xpad’, ’1’), (’ypad’, ’1’)), ’title-4’: ((’si-
ze’, ’small’), (’underline’, ’single’), (’foreground’, ’#000066’)), ’war-
ning’: ((’background’, ’#ffffdd’),), ’caution’: ((’back-
ground’, ’#ffffdd’),), ’title-3’: ((’size’, ’small’), (’weight’, ’bold’), (’
reground’, ’#000066’)), ’title-0’: ((’size’, ’xx-large’), (’weight’, ’bold’)
reground’, ’#000066’)), ’title-1’: ((’size’, ’large’), (’weight’, ’bold’), (
reground’, ’#000066’)), ’image-link’: (), ’calltip-type’: ((’font_family’, ’
poc’: ((’font_family’, ’sans’),), ’hint’: ((’background’, ’#ffffdd’),), ’adm
title’: ((’weight’, ’bold’),), ’tip’: ((’background’, ’#ffffdd’),), ’li-
teral’: ((’foreground’, ’#227722’), (’weight’, ’bold’)), ’no-
te’: (), ’field’: ((’weight’, ’bold’),), ’emphasis’: ((’sty-
le’, ’italic’),), ’calltip-class-symbol’: ((’font_family’, ’sans’), (’weight
reground’, ’#0000ff’)), ’attention’: ((’background’, ’#ddddff’),), ’calltip-
def-symbol’: ((’font_family’, ’sans’), (’weight’, ’bold’), (’fo-
reground’, ’#007f7f’)), ’link’: ((’underline’, ’single’), (’fo-
reground’, ’#3333ff’)), ’strong’: ((’weight’, ’bold’), (’fore-
ground’, ’#000066’)), ’marked-list-items’: ((’weight’, ’bold’), (’fo-
reground’, ’#ff3333’)), ’calltip-def’: ((’font_family’, ’sans’), (’weight’,
reground’, ’#00007f’)), ’calltip-doc’: ((’font_family’, ’sans’),), ’default’
header’: ((’weight’, ’bold’),), ’transition’: ((’justificati-
on’, ’left’),), ’calltip-arg’: ((’font_family’, ’sans’),), ’calltip-
arg-current’: ((’font_family’, ’sans’), (’background’, ’#ffbbbb’)), ’cap-
tion’: ((’style’, ’italic’),), ’error’: ((’background’, ’#ffdddd’),), ’title
2’: ((’size’, ’medium’), (’weight’, ’bold’), (’fore-
ground’, ’#000066’)), ’navigation’: ((’foreground’, ’#909090’), (’sty-
le’, ’italic’))}
```

# Dateien

### Default Directory Policy

Defines how Wing determines the starting directory to use when prompting for a file name: Either based on location of the resource at current focus, location of the current project, the last directory visited for file selection, the current directory at startup (or selected since), or always the specific fixed directory entered here.

Interner Name:

```
main.start-dir-policy
```

Daten Spezifikation:

```
[tuple Länge 2 von: [current-project, current-directory, recent-
directory, current-focus, selected-directory], <type str>]
```

Standardeinstellung:

```
('current-resource', '')
```

**Stil des Titels**

Format, dass für den Titel von Source-Dateien verwendet wird: Verwenden Sie „Basisname“, um nur den Dateinamen anzuzeigen, „Relativen Pfad voranstellen“, um teilweise den relativen Pfad von der Projektdateiposition zu nutzen, „Relativen Pfad anhängen“, um teilweise den relativen Pfad von der Projektdateiposition nach dem Basisdateinamen anzuhängen, „Vollen Pfad voranstellen“, um den vollen Pfad zu nutzen oder „Vollen Pfad anhängen“, um den Vollpfad nach dem Dateinamen anzuhängen.

Interner Name:

```
gui.source-title-style
```

Daten Spezifikation:

```
[append-relative, basename, prepend-fullpath, append-
fullpath, prepend-relative]
```

Standardeinstellung:

```
append-relative
```

**Im Werkzeug-Tipps immer vollständigen Pfad verwenden**

Auf Wahr setzen, um immer den vollständigen Pfad eines Dateinamens in den Werkzeug-Tipps anzuzeigen, die von den Editor-Reitern und Dateiauswahl-Menüs angezeigt werden. Wenn es auf Falsch gesetzt wird, wird stattdessen der konfigurierte Source-Titelstil verwendet.

Interner Name:

```
gui.full-path-in-tooltips
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
True
```

**Standard-Kodierung**

Die Standard-Kodierung, die für Textdateien, die im Source-Editor geöffnet sind, und für andere Werkzeuge verwendet wird, wenn beim Lesen der Datei keine Kodierung für die Datei bestimmt werden kann. Andere Kodierungen können auch probiert werden. Dies stellt auch die Kodierung ein, die für neu erstellte Dateien verwendet wird.

Interner Name:

```
edit.default-encoding
```

Daten Spezifikation:

```
[None oder [Chinesisch (PRC) hz, Türkisch cp1026, Japa-
nisch shift-jisx0213, Koreanisch johab, Chinesisch (ROC) big5, Grie-
chisch cp869, Russisch koi8-r, Arabisch cp1256, Japanisch iso-
2022-jp-2, Westeuropa cp1140, Chinesisch (PRC) gbk, Hebrä-
isch cp424, Zentral- und Osteuropa cp852, Westeuropa cp850, Un-
icode (UTF-16, little endian) utf-16-le, Baltische Spra-
chen iso8859-13, Chinesisch (ROC) cp950, Esperanto und Mal-
tesisch iso8859-3, Nordische Sprachen iso8859-10, Ukrai-
nisch koi8-u, Hebräisch iso8859-8, USA, Australien, Neu Zea-
land, Sud Afrika cp437, Chinesisch (PRC) gb18030, Icelan-
dic mac-iceland, USA, Kanada und andere cp037, Baltische Spra-
chen iso8859-4, Vietnamesisch cp1258, Urdu cp1006, Japa-
nisch shift-jis, Chinesisch (PRC) big5hkscs, Westeuropa mac-
roman, Thailändisch cp874, Koreanisch iso-2022-kr, Hebrä-
isch cp1255, Kyrillische Sprachen mac-cyrillic, Japanisch euc-
jis-2004, Japanisch iso-2022-jp-1, Griechisch iso8859-7, Dä-
nisch, Norwegisch cp865, Japanisch iso-2022-jp-ext, Grie-
chisch cp875, Arabisch cp864, Westeuropa iso8859-15, Sys-
temvoreinstellung (ISO-8859-1), Japanisch iso-2022-jp-
3, Japanisch euc-jisx0213, Japanisch shift-jis-2004, Ky-
rillische Sprachen iso8859-5, Arabisch iso8859-6, Japa-
```

```
nisch iso-2022-jp, Unicode (UTF-16, big endian) utf-16-
be, Baltische Sprachen cp1257, Portugiesisch cp860, Zentral-
und Osteuropa cp1250, Türkisch cp1254, Westeuropa latin-
1, Koreanisch cp949, Baltische Sprachen cp775, Chine-
sisch (PRC) gb2312, Japanisch cp932, None, Japanisch iso-
2022-jp-2004, Japanisch euc-jp, Keltische Sprachen iso8859-
14, Westeuropa cp1252, Hebräisch cp862, Kyrillische Spra-
chen cp855, Griechisch mac-greek, Unicode (UTF-8) utf-8, Is-
ländisch cp861, Zentral- und Osteuropa iso8859-2, Tür-
kisch iso8859-9, Englisch ascii, Unicode (UTF-7) utf-7, Tür-
kisch cp857, Hebräisch cp856, Zentral- und Osteuropa mac-
latin2, Kanadisches Englisch/Französisch cp863, Westeu-
ropa cp500, Türkisch mac-turkish, Griechisch cp737, Ky-
rillische Sprachen cp1251, Unicode (UTF-16) utf-16, Grie-
chisch cp1253]]
```

Standardeinstellung:

```
None
```

### Zeilenende für Neue Dateien

Zu verwendendes voreingestelltes Zeilenende: Entweder „lf", „cr" oder „crlf" für jeden Eintrag. Beachten Sie, dass Wing bestehende Zeilenenden in nicht-leeren Dateien abstimmt und die Einstellung nur verwendet, wenn eine Datei keine Zeilenende-Zeichen enthält.

Interner Name:

```
edit.new-file-eol-style
```

Daten Spezifikation:

```
[lf, cr, crlf]
```

Standardeinstellung:

```
lf
```

### Dateizusatz für Neue Dateien

Voreingestellter Dateizusatz für neu erstellte Dateien

Interner Name:

```
edit.new-file-extension
```

Daten Spezifikation:

```
<type str>
```

Standardeinstellung:

```
.py
```

**Maximale Anzahl der letzten Dokumente**

Maximale Anzahl der Einträge, die in Letzte Menüs angezeigt werden.

Interner Name:

```
gui.max-recent-files
```

Daten Spezifikation:

```
[von 3 bis 200]
```

Standardeinstellung:

```
20
```

- **Dateiarten**

**Extra-Dateiarten**

Dies ist eine Konvertierung von Dateizusätzen oder Wildcards zu Mime-Typen. Dies fügt zusätzliche Konvertierungen von Dateitypen zu denen, die in Wing IDE erstellt werden, hinzu. Dateizusätze können allein ohne Punkt oder Wildcard angegeben werden, z. B. „xcf“ oder sie können Wildcards verwenden, die „*“ und/oder „?“ enthalten, z. B. „Makefile*“. Der Mime-Typ, der für Python-Dateien verwendet wird, lautet „text/x-python“.

Interner Name:

```
main.extra-mime-types
```

Daten Spezifikation:

```
[dict; keys: <type str>, Werte: [text/x-sql, text/x-
pov, text/x-ave, text/x-pl-sql, text/x-bash, text/x-lua-
source, text/x-eiffel, text/x-vxml, text/xml, text/x-
errorlist, text/x-php-source, text/x-dos-batch, text/x-
bullant, text/x-baan, text/x-python, text/x-nncrontab, text/x-
mmixal, text/postscript, text/x-javascript, text/x-fortran, text/x-
xcode, text/x-escript, text/x-lisp, text/x-makefile, text/x-
diff, text/x-ms-idl, text/x-cpp-source, text/x-asm, text/x-
ruby, text/x-ada, text/x-nsis, text/x-idl, text/x-scriptol, text/x-
perl, text/x-java-source, text/x-docbook, text/x-rc, text/x-c-
source, text/plain, text/x-lout, text/x-matlab, text/html, application/x-
tex, text/x-tcl, text/x-vb-source, text/x-pascal, text/x-
yaml, text/x-conf, text/x-ms-makefile, text/x-properties, text/css]]
```

Standardeinstellung:

```
{}
```

**Datei-Sets**

Definiert Datei-Sets durch die Bestimmung von Filtern, die auf Dateinamen angewendet werden, um deren Einbeziehung oder Ausschluss aus einem größeren Set festzulegen (wie gescannte Laufwerkdateien oder alle Projektdateien).

Jedes Datei-Set hat einen Namen und enthält eine Liste mit den Einschlusskriterien und eine Liste mit den Ausschlusskriterien. Die Kriterien können entweder eine Wildcard auf den Dateinamen, eine Wildcard auf den Verzeichnisnamen oder ein Mime-Typ-Name sein.

Nur ein einziges Kriterium muss übereinstimmen, um die Einbeziehung oder den Ausschluss zu bewirken. Ausschlusskriterien haben Vorrang gegenüber Einschlusskriterien, so dass eine Übereinstimmung mit einem Ausschlusskriterium die Datei immer von dem Set ausschließt. Datei-Sets werden für begrenzende Suchen, beim Hinzufügen von Projektdateien und für andere Operationen an Dateisammlungen verwendet.

Interner Name:

```
main.file-sets
```

Daten Spezifikation:

```
[dict; keys: <type str>, Werte: [tuple Länge 2 von: [tuple von: [tuple Länge 2 von: [wildcard-filename, wildcard-directory, mime-type], <type str>]], [tuple von: [tuple Länge 2 von: [wildcard-filename, wildcard-directory, mime-type], <type str>]]]]
```

Standardeinstellung:

```
{u'All Source Files': ((), (('wildcard-filename', '*.o'), ('wildcard-filename', '*.obj'), ('wildcard-filename', '*.a'), ('wildcard-filename', '*.lib'), ('wildcard-filename', '*.so'), ('wildcard-filename', '*.dll'), ('wildcard-filename', '*.exe'), ('wildcard-filename', '*.ilk'), ('wildcard-filename', '*.pdb'), ('wildcard-filename', '*.pyc'), ('wildcard-filename', '*.pyo'), ('wildcard-filename', '*.pyd'), ('wildcard-filename', 'core'), ('wildcard-filename', '*.bak'), ('wildcard-filename', '*.tmp'), ('wildcard-filename', '*.temp'), ('wildcard-filename', '*-old'), ('wildcard-filename', '*.old'), ('wildcard-filename', '*.wpr'), ('wildcard-filename', '*.wpu'), ('wildcard-filename', '*.zip'), ('wildcard-filename', '*.tgz'), ('wildcard-filename', '*.tar.gz'), ('wildcard-filename', '*~'), ('wildcard-filename', '#*#'), ('wildcard-filename', '.#*'), ('wildcard-filename', '*.svn-base'), ('wildcard-directory', 'CVS'), ('wildcard-directory', '.svn'), ('wildcard-directory', '.xvpics'))), u'HTML and XML Files': ((('mime-type', 'text/html'), ('mime-type', 'text/xml')), (('wildcard-directory', 'CVS'), ('wildcard-directory', '.svn'), ('wildcard-directory', '.xvpics'))), u'C/C++ Files': ((('mime-type', 'text/x-c-source'), ('mime-type', 'text/x-cpp-source')), (('wildcard-directory', 'CVS'), ('wildcard-directory', '.svn'), ('wildcard-directory', '.xvpics'))), u'Python Files': ((('mime-type', 'text/x-python'),), (('wildcard-directory', 'CVS'), ('wildcard-directory', '.svn'), ('wildcard-directory', '.xvpics')))}
```

- **Neu laden**

### Externe Prüffrequenz

Zeit in Sekunden, die die Frequenz angibt, mit welcher das IDE das Laufwerk auf extern geänderte Dateien überprüfen sollte. Auf 0 einstellen, um es vollständig zu deaktivieren.

Interner Name:

```
cache.external-check-freq
```

Daten Spezifikation:

```
<type float>, <type int>
```

Standardeinstellung:

```
5
```

### Neuladen wenn unverändert

Wählt Aktion, um Dateien auszuführen, die extern geändert wurden, aber innerhalb des IDEs unverändert sind. Dabei ist „Automatisch Neuladen“,um diese Dateien automatisch wieder zu laden, „Neuladen sofort Anfordern“, um nach Feststellung über eine Dialogbox zu fragen, „Neuladen beim Bearbeiten anfordern“, um nur zu fragen, wenn die unveränderte Datei innerhalb des IDE nachträglich bearbeitet wurde, oder „Niemals Neuladen“, um externe Änderungen zu ignorieren (obwohl Sie trotzdem noch gewarnt werden, wenn sie versuchen, eine extern geänderte Datei zu überspeichern)

Interner Name:

```
cache.unchanged-reload-policy
```

Daten Spezifikation:

```
[never-reload, auto-reload, request-reload, edit-reload]
```

Standardeinstellung:

```
request-reload
```

### Neuladen wenn geändert

Wählt Aktion, um Dateien auszuführen, die sowohl extern als auch innerhalb des IDE geändert wurden. Dabei ist „Neuladen sofort Anfordern“, um nach Feststellung über eine Dialogbox anzufragen, „Neuladen beim Bearbeiten anfordern“, um zu fragen, ob die Datei weiter bearbeitet wurde, oder „Niemals Neuladen“, um externe Änderungen zu ignorieren (obwohl sie trotzdem immer gewarnt werden, wenn Sie versuchen, eine extern geänderte Datei zu überspeichern.)

Interner Name:

```
cache.changed-reload-policy
```

Daten Spezifikation:

```
[never-reload, request-reload, edit-reload]
```

Standardeinstellung:

```
request-reload
```

- **Projekte**

**Letztes Projekt automatisch wieder öffnen**

Kontrolliert, ob das letzte Projekt mangels eines anderen Projektes in der Befehlszeile beim Starten wieder geöffnet wird.

Interner Name:

```
main.auto-reopen-last-project
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

**Dateien mit Projekt schließen**

Kontrolliert, ob in einem Editor geöffnete Dateien geschlossen werden, wenn ein Projektdatei geschlossen wird

Interner Name:

```
proj.close-also-windows
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

### Standardtyp

Kontrolliert den Typ der Projektdatei, der standardmäßig für neue Projekte verwendet wird: „Normal“ for reguläres Einzeldatei-Format mit dem Zusatz .wpr, und „Gemeinsam“ für geteiltes Format, bei dem die .wpr Datei gemeinsame Projektinformationen enthält, die in ein gemeinsames Revisionskontroll-System eingecheckt werden können und die .wpu Datei enthält nutzer-spezifische Informationen, wie Position der Haltepunkte. Das ist nützlich, um bei einem Projekt mit vielen Entwicklern Revisionskontroll-Kriege zu vermeiden.

Interner Name:

```
proj.file-type
```

Daten Spezifikation:

```
[shared, normal]
```

Standardeinstellung:

```
normal
```

### Automatisch hinzufügen

Kontrolliert, ob Dateien automatisch zum aktuellen Projekt hinzugefügt werden. Entweder alle Dateien hinyufügen, die auf dem Laufwerk gespeichert werden, wenn das Projekt geöffnet ist, nur neu erstellte Dateien hinzufügen oder keine Dateien automatisch hinzufügen.

Interner Name:

```
proj.auto-add-policy
```

Daten Spezifikation:

```
[all-saved, all-new, never]
```

Standardeinstellung:

```
never
```

**Projekte als Text öffnen**

Kontrolliert, ob Projektdateien als Projekt oder als Text geöffnet werden, wenn sie vom Menü Datei geöffnet werden. Dies beeinflußt nicht das Öffnen vom Menü Projekt.

Interner Name:

```
gui.open-projects-as-text
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

- **Externes anzeigen**

**Befehle der Dateianzeige**

Nur Linux: Die verwendeten Befehle, um lokale Laufwerkdateien, die aus dem Hilfemenü ausgewählt sind, oder Projektdateien, die für die externe Anzeige ausgewählt sind, anzuzeigen oder zu bearbeiten. Dies ist eine Abbildung von Mime-Typen in einer Liste von Bildschirmbefehlen, jeder Bildschirmbefehl wird im Auftrag der Liste getestet, bis einer funktioniert. Der Mime-Typ „*“ kann genutzt werden, um ein generelles Betrachterprogramm einzustellen, wie einen Web-Browser. Verwenden Sie %s, um den Dateinamen in den Befehlszeilen zu platzieren. Wenn nicht angegeben, verwendet Wing das konfigurierte URL-Betrachterprogramm in der Umgebung (bestimmt duch die Umgebungsvariable BROWSER oder duch Suchen des Pfades für allgemeine Browser). In Windows und OS X wird stattdessen das systemweit konfigurierte Standardbetrachterprogramm für den Dateitypen genutzt, so dass diese Einstellung ignoriert wird.

Interner Name:

```
gui.file-display-cmds
```

Daten Spezifikation:

```
[dict; keys: <type str>, Werte: [list von: <type str>]]
```

Standardeinstellung:

```
{}
```

### Befehle der URL-Anzeige

Nur Linux: Die verwendeten Befehle, um URLs anzuzeigen. Dies ist eine Abbildung von Protokolltypen in einer Liste von Bildschirmbefehlen, jeder Bildschirmbefehl wird im Auftrag der Liste getestet, bis einer funktioniert. Das Protokoll „*“ kann genutzt werden, um ein generelles Betrachterprogramm, wie einen Mehrfach-Protokoll-Web-Browser, einzustellen. Verwenden Sie %s, um die URL in den Befehlszeilen zu platzieren. Wenn nicht angegeben, verwendet Wing das konfigurierte URL-Betrachterprogramm in der Umgebung (bestimmt durch die Umgebungsvariable BROWSER oder durch Suchen des Pfades für allgemeine Browser). In Windows und OS X wird stattdessen der systemweit konfigurierte Web-Browser genutzt, so dass diese Einstellung ignoriert wird.

Interner Name:

```
gui.url-display-cmds
```

Daten Spezifikation:

```
[dict; keys: <type str>, Werte: [list von: <type str>]]
```

Standardeinstellung:

```
{}
```

## Editor

### Breite der Zeilennummernspalte

Breite der Zeilennummernsplate (0 zum verstecken). Wenn weniger als 10, wird die Einstellung als Nummer von Ziffern benutzt. Wenn 10 oder mehr, wird sie als Bildpunktbreite benutzt.

Interner Name:

```
edit.lineno-column-width
```

Daten Spezifikation:

```
<type int>
```

Standardeinstellung:

```
0
```

**Caret-Breite**

Breite des blinkenden Einfügungscarets im Editor in Pixeln. Zur Zeit auf einen Wert zwischen 1 und 3 begrenzt.

Interner Name:

```
edit.caret-width
```

Daten Spezifikation:

```
[von 1 bis 3]
```

Standardeinstellung:

```
1
```

**Leerraum anzeigen**

Auf wahr setzen, um Leeraum mit sichtbaren Zeichen standardmäßig anzuzeigen

Interner Name:

```
edit.show-whitespace
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

### Zeilenende anzeigen

Auf wahr setzen, um das Zeilenende mit sichtbaren Zeichen standardmäßig anzuzeigen

Interner Name:

```
edit.show-eol
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

### Grundsatz für die Wiederverwendung von Teilungen

Grundsatz für die Wiederverwendung von Teilungen in Editoren wenn neue Dateien geöffnet werden. Entweder in aktueller Teilung oder in benachbarter Teilung öffnen. Dies hat nur Auswirkungen wenn mehr als eine Editor-Teilung sichtbar ist.

Interner Name:

```
gui.split-reuse-policy
```

Daten Spezifikation:

```
[current, adjacent]
```

Standardeinstellung:

```
current
```

### Texteingabemethode

Texteingabemethode. Dieses wird haubtsächlich für nicht-West Europäische Sprachen benutzt.

Interner Name:

```
edit.gtk-input-method
```

Daten Spezifikation:

```
[]
```

Standardeinstellung:

```
default
```

- **Einrückung**

**Use Indent Analysis**

Select when to use indent analysis (examination of current file contents) in order to determine tab size and indent size. Either always in all files, only in Python files, or never.

Interner Name:

```
edit.use-indent-analysis
```

Daten Spezifikation:

```
[always, never, python-only]
```

Standardeinstellung:

```
always
```

**Standard-Tabgröße**

Stellt Größe der Tabs (in Leerzeichen) ein, die in neuen Dateien verwendet werden. Beachten Sie, dass in Python-Dateien, die gemischte Leerzeichen- und Tab-Einrückungen beinhalten, die Tabgröße immer zwingend acht Leerzeichen erfordert. Verwenden Sie den Einrückungsmanager, um Einrückungen in bestehenden Dateien zu ändern.

Interner Name:

```
edit.tab-size
```

Daten Spezifikation:

```
[von 0 bis 80]
```

Standardeinstellung:

```
8
```

**Standard-Einrückungsgröße**

Stellt die Größe eines Einzugs (in Leerzeichen) ein, die in neuen Dateien verwendet werden. Dies wird in nicht-leeren Dateien außer Kraft gesetzt, entsprechend den aktuellen Inhalten der Datei. In Dateien mit Nur-Tab Einrückung wird diese Einstellung automatisch geändert so das es ein Veilfach der Tabgrösse wird. Verwenden Sie den Einrückungsmanager, um Einrückungen in bestehenden Dateien zu ändern.

Interner Name:

```
edit.indent-size
```

Daten Spezifikation:

```
[von 0 bis 80]
```

Standardeinstellung:

```
2
```

**Standard-Einrückungsstil**

Stellt die Art der Einrückung ein, wie sie in neuen Dateien verwendet wird. Dies wird in nicht-leeren Dateien außer Kraft gesetzt, entsprechend den aktuellen Inhalten der Datei. Verwenden Sie den Einrückunsmanager, um Einrückungen in bestehenden Dateien zu ändern. Die Wahlmöglichkeiten für den Einrückungsmanager sind „Nur-Tabs“ für Nur-Tabs, „Nur-Leerzeichen“ für Nur-Leerzeichen oder „Gemischt“, um einen Tab zu nutzen, wann immer Leerzeichen in Tabgröße gesehen werden.

Interner Name:

```
edit.indent-style
```

Daten Spezifikation:

```
[mixed, spaces-only, tabs-only]
```

Standardeinstellung:

```
spaces-only
```

**Einrückungslinien anzeigen**

Auf wahr setzen, um die Einrückungslinien standardmäßig anzuzeigen

Interner Name:

```
edit.show-indent-guides
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

**Automatisch einrücken**

Controls when Wing automatically indents when return or enter is typed.

Interner Name:

```
edit.auto-indent
```

Daten Spezifikation:

```
[0, 1, blank-only]
```

Standardeinstellung:

```
1
```

**Warnungen bei Nichtübereinstimmungen anzeigen**

Einrückungwarnung anzeigen nach wählen von ein Einrückungsstil das nicht zu existierende Einrückung im Datei passt. Diese Einstellung wird nur für nicht-Python Dateien benutzt.

Interner Name:

```
edit.show-non-py-indent-warning
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
True
```

- **Zeilenumbruch**

**Lange Zeilen umbrechen**

Auf wahr setzen, um lange Source-Zeilen in der Editor-Ansicht umzubrechen.

Interner Name:

```
edit.wrap-lines
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

**Kantenmarkierungen**

Tuple, das definiert, wie Kantenmarkierungen angezeigt werden (Modus, Spalte, Farbe), wobei der Modus 0 ist, um Markierungen abzuschalten, 1, um eine Zeile anzuzeigen oder

2, um Text, der sich über die Kante ausdehnt, hervorzuheben; Spalte ist die Spalte, an der die Markierung gezeichnet wird, wenn an; und Farbe ist die Farbe für die Markierung (r,g,b) Tuple mit Werten von 0x00 bis 0xff: (0xff,0xff,0xff) ist weiß.

Interner Name:

```
edit.show-edge-markers
```

Daten Spezifikation:

```
[tuple Länge 3 von: [0, 1, 2], [von 0 bis 10000], [tuple Län-
ge 3 von: [von 0 bis 255], [von 0 bis 255], [von 0 bis 255]]]
```

Standardeinstellung:

```
(0, 80, (251, 8, 8))
```

### Zeilenumbruch neu formatieren

Spalte, an der Text durch Befehle, die den Text automatisch neu anordnen, umgebrochen werden sollte

Interner Name:

```
edit.text-wrap-column
```

Daten Spezifikation:

```
<type int>
```

Standardeinstellung:

```
77
```

- **Falten**

### Falten aktivieren

Auf wahr stellen, um strukturelles Falten im Source-Code zu ermöglichen, falsch, um zu deaktivieren

Interner Name:

```
edit.enable-folding
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

**Zeilenmodus**

Auf „Oben Erweitert“, „Unten Erweitert“, „Oben Zusammengeklappt“, „UntenZusammengeklappt“ oder „Keine“ setzen, um anzuzeigen, wo Faltezeilen gezeigt werden und ob sie über oder unter der Zeile sind, an der der Faltepunkt platziert ist.

Interner Name:

```
edit.fold-line-mode
```

Daten Spezifikation:

```
[above-collapsed, above-expanded, none, below-collapsed, below-
expanded]
```

Standardeinstellung:

```
below-collapsed
```

**Indikatorstil**

Auf 0 setzen, um Pfleilindikatoren zu verwenden, 1, um Plus/Minus-Indikatoren zu nutzen, 2 für gerundente Baumindikatoren und 3 für die Verwendung von quadratischen Baumindikatoren.

Interner Name:

```
edit.fold-indicator-style
```

Daten Spezifikation:

```
[von 0 bis 3]
```

Standardeinstellung:

```
1
```

- **Auto-Vervollständigung**

**Auto-Vervollständiger automatisch anzeigen**

Kontrolliert ob der Auto-Vervollständiger automatisch aufgeschlagen wird während getippt wird. Wenn deaktiviert benutzen Sie Vervollständiger Aufschlagen im Source Menü.

Interner Name:

```
edit.autocomplete-autoshow
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

**Verzögerung des Auto-Vervollständigers**

Abschaltung in Sekunden vom letzten Tastendruck, nach welchem der Auto-Vervollständiger automatisch aufgeschlagen wird. Wenn 0.0, wird der Auto-Vervollständiger gleich aufgeschlagen.

Interner Name:

```
edit.autocomplete-delay
```

Daten Spezifikation:

```
<type int>, <type float>
```

Standardeinstellung:

```
0.0
```

**Abschaltung des Auto-Vervollständigers**

Abschaltung in Sekunden vom letzten Tastendruck, nach welchem der Auto-Vervollständiger automatisch versteckt wird. Wenn 0.0, wird der Auto-Vervollständiger nicht abgeschalten.

Interner Name:

```
edit.autocomplete-timeout
```

Daten Spezifikation:

```
<type int>, <type float>
```

Standardeinstellung:

```
0
```

**Vervollständigungstasten**

Kontrolliert welche Tasten das gewählte Feld im Auto-Vervollständiger zum Editor vervolöständigt. Benutzen Sie die Umschalt und Strng Tasten während Klicken um mehrere werte gleichzeitig zu wählen.

Interner Name:

```
edit.autocomplete-keys
```

Daten Spezifikation:

```
[tuple von: [f1, f3, return, space, tab, f12, f10]]
```

Standardeinstellung:

```
('tab',)
```

### Vervollständigungsmodus

Kontrolliert wie der Auto-Vervollständiger Text im Editor einfügt: Entweder am Cursor einfach einfügen, oder existierendes Symbol vor und nach dem Cursor ersetzen.

Interner Name:

```
edit.autocomplete-mode
```

Daten Spezifikation:

```
[insert, replace]
```

Standardeinstellung:

```
insert
```

### Groß- und Kleinschreibung beachten

Kontrolliert ob übereinstimmung im Auto-Vervollständiger Groß- und Kleinschreibung beachtet. In jeden Fall wird die korrecte Groß- und Kleinschreibung wird immer am einfügen im Editors benutzt.

Interner Name:

```
edit.autocomplete-case-insensitive
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
True
```

- **Drucken**

### Schriftart

(Nur Posix) Stellt den Schriftnamen ein, der für das Drucken von Python-Dateien verwendet wird. Entweder Courier, Helvetica oder Times-Roman.

Interner Name:

```
edit.print-font
```

Daten Spezifikation:

```
[Times-Roman, Helvetica, Courier]
```

Standardeinstellung:

```
Courier
```

**Schriftgröße**

(Nur Posix) Stellt die Schiftgröße ein, die für das Drucken von Python-Dateien verwendet wird.

Interner Name:

```
edit.print-size
```

Daten Spezifikation:

```
[von 0 bis 120]
```

Standardeinstellung:

```
10
```

**Papier**

(Nur Posix) Stellt das Papierformat für den Druck ein. Entweder US Letter, Legal, A3, A4, A5, B4 oder B5

Interner Name:

```
edit.print-paper
```

Daten Spezifikation:

```
[A3, A5, Legal, Letter, A4]
```

Standardeinstellung:

```
Letter
```

### Druckbefehl senden

(Nur Posix) Stellt den Befehl ein, der genutzt wird, um die Postskript-Ausgabe, die von Wing's Druckservice produziert wird, zu senden. Format ist Text mit eingebetteten %s, um anzuzeigen, wo der Name der gedruckten Datei eingefügt werden sollte. Auf None stellen, um die internen Voreinstellungen zu verwenden.

Interner Name:

```
edit.print-spool-cmd
```

Daten Spezifikation:

```
[einer von: None, <type str>]
```

Standardeinstellung:

```
None
```

### Python als Text drucken

(Nur Posix) Auf wahr setzen, um Python-Dateien schneller, aber ohne Syntax-Markierung zu drucken. Andernfalls wird der interne Python-Druckservice verwendet.

Interner Name:

```
edit.print-python-as-text
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

**Befehl für Textdruck**

(Nur Posix) Stellt den Befehl ein, der erteilt wird, um non-Python-Textdateien zu drucken. Das Format ist Text mit eingebetteten %s , um anzuzeigen, wo der Name der gedruckten Datei eingefügt werden sollte

Interner Name:

```
edit.text-print-cmd
```

Daten Spezifikation:

```
<type str>
```

Standardeinstellung:

```
enscript -E %s
```

- **Erweitert**

**Automatische Klammernsuche**

Auf Wahr setzen, um automatisch Klammern neben dem Cursor oder wenn sie getippt werden zu suchen.

Interner Name:

```
edit.auto-brace-match
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

**Schwelle für vorübergehende Dateien**

Maximale Anzahl von vorübergehenden (nicht-sticky) Editoren, die zur gleichen Zeit geöffnet bleiben können, zusätzlich zu denen, die auf dem Bildschirm sichtbar sind.

Interner Name:

```
gui.max-non-sticky-editors
```

Daten Spezifikation:

```
<type int>
```

Standardeinstellung:

```
1
```

**Auswahlgrundsatz**

Dies ist eine Übersicht von Aktionen zum Grundsatz für das Belassen eines ausgewählten Bereiches, nachdem die Aktion stattfindet. Mögliche Aktionen sind „indent-region“, „outdent-region“, „indent-to-match“, „comment-out-region“, und „uncomment-out-region“. Mögliche Vorgehensweisen für jede sind „always-select“, was immer eine Auswahl lässt, „retain-select“, was nur ein Auswahl lässt, wenn es eine gibt, mit der begonnen werden kann und „never-select“, was niemals eine Auswahl lässt.

Interner Name:

```
edit.select-policy
```

Daten Spezifikation:

```
[dict; keys: [(u'Einr\xfccken zum Anpassen', 'indent-to-
match'), (u'Ausr\xfcckungsbereich', 'outdent-region'), (u'Kommentar im Be-
reich aufheben', 'uncomment-out-region'), (u'Einr\xfcckungsbereich', 'indent
region'), (u'Bereich auskommentieren', 'comment-out-
region')], Werte: [(u'Immer ausw\xe4hlen', 'always-select'), (u'Niemals ausw
select'), (u'Ausw\xe4hlen beibehalten', 'retain-select')]]
```

Standardeinstellung:

```
{'uncomment-out-region': 'retain-select', 'outdent-region': 'retain-
select', 'comment-out-region': 'retain-select', 'indent-
region': 'retain-select', 'indent-to-match': 'retain-select'}
```

**Einfügen mit mittlerer Maustaste**

Text von der Zwischenablage zum Editor einfügen wenn die Mittelmaustaste gedrückt wird. Diese Einstellung deaktivieren um probleme zu vermeiden mit Mausräder die auch als Mittelmaustaste dienen.

Interner Name:

```
edit.middle-mouse-paste
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
True
```

# Debugger

### Anzeigemodus für Ganzzahlen

Dies stellt die Ansichtsart für ganzzahlige Werte entweder auf „Dezimal", „Hexal", oder „Oktal" ein.

Interner Name:

```
debug.default-integer-mode
```

Daten Spezifikation:

```
[dec, hex, oct]
```

Standardeinstellung:

```
dec
```

### Dateien automatisch speichern

Kontrolliert, ob alle bearbeiteten Dateien automatisch vor einem Debug-Durchlauf oder vor der Ausführung einer Datei oder eines Build-Prozesses gespeichert werden.

Interner Name:

```
gui.auto-save-before-action
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

**Nicht synchronisierte Dateien ignorieren**

Kontrolliert, ob Wing ungespeicherte Dateien vor einem Debug-Durchlauf oder vor der Ausführung einer Datei oder eines Build-Prozesses ignoriert.

Interner Name:

```
gui.ignore-unsaved-before-action
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

**Source von Werkzeugen aufschlagen**

Kontrolliert, ob der Debugger Source-Dateien aufschlägt, um Exception-Positionen, die angetroffen werden, wenn im Debug-Test und anderen Debug-Werkzeugen gearbeitet wird, anzuzeigen.

Interner Name:

```
debug.raise-from-tools
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

### Standard-Beobachtungsstil

Setzt den Verfolgungsstil, der verwendet wird, wenn ein Wert doppelt angeklickt wird, um ihn zu beobachten: Verwenden Sie „symbolic“, um nach symbolischem Namen zu verfolgen, „parent-ref“, um Parents nach Objektverweis und Attribute nach Namen zu verfolgen, und „ref“, um unter Verwendung eines Objektverweises direkt auf den Wert zu verfolgen.

Interner Name:

```
debug.default-watch-style
```

Daten Spezifikation:

```
[ref, parent-ref, symbolic]
```

Standardeinstellung:

```
symbolic
```

### Zeilenschwelle

Definiert die Schwelle für Zeichenlängen unter der ein Wert immer auf einer einzelnen Zeile angezeigt wird, selbst wenn der Wert ein komplexer Typ, wie ein list oder map, ist.

Interner Name:

```
debug.line-threshold
```

Daten Spezifikation:

```
<type int>
```

Standardeinstellung:

```
65
```

- **Exceptions**

### Exceptions Berichten

Steuert, wie Wing Exceptions, die in Ihrem Debug-Prozess angetroffen werden, berichtet. Standardmäßig versucht Wing vorherzusagen, welche Exceptions unbehandelt sind und hält sofort an, wenn unbehandelte Exceptions auftreten. Alternativ kann Wing auch an jeder Exception (auch wenn behandelt) sofort anhalten, wenn diese auftreten, oder Wing kann fatale Exception beim Beenden des Debug-Prozesses berichten. Im letzteren Fall versucht Wing, vor dem Beenden desDebug-Prozesses zu stoppen, oder zumindestens eine Rückverfolgung nach dem Beendenzu ermöglichen. Allerdings kann eine oder beide Möglichkeiten scheitern, wenn Sie mit extern gestarteten Debug-Prozessen arbeiten. In diesem Fall empfehlen wir Ihnen, den Exception-Berichtmodus „Sofort, wenn scheinbar unbehandelt" zu verwenden.

Interner Name:

```
debug.exception-mode
```

Daten Spezifikation:

```
[unhandled, always, never]
```

Standardeinstellung:

```
unhandled
```

### Nie Anzeigen

Die Namen von den built-in Exceptions die der Debugger nie berichtet. Diese Liste setzt die Liste von Exceptions die immer berichtet werden und auch das Exception Berichtmodi außer Kraft, aber nicht im Exception Berichtmodi „immer sofort" wo alle Exceptions sofort berichtet werden.

Interner Name:

```
debug.never-stop-exceptions
```

Daten Spezifikation:

```
[tuple von: <type str>]
```

Standardeinstellung:

```
['SystemExit']
```

**Immer Anzeigen**

Die Namen von den built-in Exceptions im Debugprozess die (beinah) immer berichtet werden. Diese Exceptions werden nicht berichtet nur wenn sie im selben Stack-Frame wo sie auftreten auch mit der genaue Exceptionklasse abgefangen werden.

Interner Name:

```
debug.always-stop-exceptions
```

Daten Spezifikation:

```
[tuple von: <type str>]
```

Standardeinstellung:

```
['AssertionError', 'NameError', 'UnboundLocalError']
```

- **I/O**

**Externe Konsole verwenden**

Bestimmt, ob für die Eingabe/Ausgabe des Debug-Prozesses das integrierte I/O-Feld oder ein externes Terminalfenster verwendet wird. Verwenden Sie ein externes Fenster, wenn Ihr Debug-Prozess von Einzelheiten der Command-Prompt-Umgebung für Cursor-Bewegung, Farbtext, etc. abhängt.

Interner Name:

```
debug.external-console
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

### Externe Konsole wartet auf Beenden

Auf wahr einstellen, um die Konsole nach dem normalen Programmende offen zu lassen, oder falsch, um die Konsole in allen Fällen sofort zu schließen. Dies ist nur relevant, wenn mit einer externen, nativen Konsole ausgeführt wird, anstatt das integrierte Debug-I/O-Feld zu verwenden.

Interner Name:

```
debug.persist-console
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

### Externe Konsolen

Eine Liste der xterm-kompatiblen X Windows Terminalprogramme, die mit Debug-Prozessen verwendet werden, wenn mit einer externen Konsole ausgeführt wird. Jedes wird der Reihe nach getestet, bis eines gefunden wird, das existiert. Wenn nur der Name angegeben ist, wird Wing nach jedem erst in PATH und dann an wahrscheinlichen Plätzen suchen. Geben Sie den vollstänigen Pfad (beginnend mit „/“) an, um eine bestimmte Executable zu verwenden.

Interner Name:

```
debug.x-terminal
```

Daten Spezifikation:

```
[tuple von: <type str>]
```

Standardeinstellung:

```
('xterm', 'konsole', 'gnome-terminal', 'rxvt')
```

- **Datenfilter**

**Große Listenschwelle**

Definiert die Längenschwelle, über der ein list, map oder anderer komplexer Typ als zu lang betrachtet wird, um ihn im normalen Debugger anzuzeigen. Ist diese zu groß eingestellt, wird der Debugger abgeschalten (Siehe Einstellung Netzwerkabschaltung)

Interner Name:

```
debug.huge-list-threshold
```

Daten Spezifikation:

```
<type int>
```

Standardeinstellung:

```
2000
```

**Große String-Schwelle**

Definiert die Länge, über der ein String als zu groß gilt, um in in der Ansicht des Debuggers abgerufen zu werden. Ist dies zu groß eingestellt, wird der Debugger abgeschalten (siehe Einstellung Netzwerkabschaltung).

Interner Name:

```
debug.huge-string-threshold
```

Daten Spezifikation:

```
<type int>
```

Standardeinstellung:

```
64000
```

**Typen auslassen**

Definiert Typen, für die Werte niemals vom Debugger angezeigt werden.

Interner Name:

```
debug.omit-types
```

Daten Spezifikation:

```
[tuple von: <type str>]
```

Standardeinstellung:

```
('function', 'builtin_function_or_method', 'class', 'class-
obj', 'instance method', 'type', 'module', 'ufunc', 'wrap-
per_descriptor', 'method_descriptor', 'member_descriptor', 'ge-
nerator')
```

### Namen auslassen

Definiert Namen für Variablen/Tasten, deren Werte niemals vom Debugger angezeigt werden.

Interner Name:

```
debug.omit-names
```

Daten Spezifikation:

```
[tuple von: <type str>]
```

Standardeinstellung:

```
()
```

### Nicht erweitern

Definiert Typen, deren Werte niemals auf Inhalte überprüft werden sollten. Dies sind Typen, die dafür bekannt sind, abzustürzen, wenn der Debugger sie überprüft, weil sie fehlerhaften Datenwertzugriffscode enthalten. Diese Werte werden stattdessen als unlesbarer Wert mit Hex-Instanz-ID des Objektes angezeigt.

Interner Name:

```
debug.no-probe-types
```

Daten Spezifikation:

```
[tuple von: <type str>]
```

Standardeinstellung:

```
('GdkColormap', 'IOBTree')
```

- **Extern/Remote**

**Passives Hören aktivieren**

Kontrolliert, ob der Debugger passiv auf Verbindungen von einem extern gestarteten Program hört (falsch zum deaktivieren, wahr zum aktivieren). Dies sollte eingeschaltet sein, wenn das Debug-Programm nicht vom IDE gestartet wird (z.B. bei einem CGI-Skript).

Interner Name:

```
debug.passive-listen
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

**Erlaubte Hosts**

Stellt ein, welche Hosts erlaubt sind, zum Debugger zu verbinden, wenn dieser passiv auf extern gestartete Programme hört.

Interner Name:

```
debug.passive-hosts
```

Daten Spezifikation:

```
[tuple von: <type str>]
```

Standardeinstellung:

```
('127.0.0.1',)
```

**Server-Host**

Bestimmt die Netzwerkoberfläche, auf welcher der Debugger auf Verbindungen hört. Dies kann ein symbolischer Name oder eine IP-Adresse sein oder unbestimmt bleiben (verwende None), um anzuzeigen, dass der Debugger auf alle gültigen Netzwerkoberflächen auf der Maschine hören sollte. Beachten Sie, dass wenn die Debug-Sitzung innerhalb des IDE gestartet wird (mit der Schaltfläche Ausführen), verbindet es immer von der Rückkopplungsoberfläche (127.0.0.1)

Interner Name:

```
debug.network-server
```

Daten Spezifikation:

```
[None oder <type str>]
```

Standardeinstellung:

```
None
```

**Server-Port**

Determines the TCP/IP port on which the IDE will listen for the connection from the debug process. This needs to be unique for each developer working on a given host. The debug process, if launched from outside of the IDE, needs to be told the value specified here using kWingHostPort inside wingdbstub.py or by WINGDB_HOSTPORT environment variable before importing wingdbstub in the debug process.

Interner Name:

```
debug.network-port
```

Daten Spezifikation:

[von 0 bis 65535]

Standardeinstellung:

50005

**Abbildung der Dateiposition**

Definiert eine Abbildung zwischen den Remote- und lokalen Positionen der Dateien für das Host-zu-Host Debuggen. Jeder Abbildungsschlüssel ist die IP-Adresse der Remote-Position und die Abbildungswerte sind Reihen von Tuples, wobei jedes Tuple ein (remote_prefix, local_prefix) Paar ist. Dies sollte genutzt werden, wenn Dateien auf dem Remote-Host via ftp, NFS, Samba oder anderen Methoden von Master-Kopien auf dem lokalen Host aktualisiert werden, aber die Vollpfad-Dateisystem-Positionen auf den lokalen und Remote-Hosts nicht zusammenpassen.

Interner Name:

debug.location-map

Daten Spezifikation:

[dict; keys: <type str>, Werte: [None oder [list von: [tuple Länge 2 von: <type str>, <type str>]]]]

Standardeinstellung:

{'127.0.0.1': None}

**Extern Gestartete löschen**

Aktiviert oder deaktiviert den Befehl Löschen für Debug-Prozesse, die außerhalb des IDE gestartet wurden.

Interner Name:

debug.enable-kill-external

Daten Spezifikation:

<boolean: 0 oder 1>

Standardeinstellung:

```
0
```

**Gemeinsame Anhängen-Hosts**

List of host/port combinations that should be included by default in the attach request list shown with Attach to Process in the Debug menu, in addition to those that are registered at runtime. These are used primarily with externally launched debug processes, since Wing automatically shows IDE-launched processes for attach when appropriate. This value corresponds with kAttachPort configured in wingdbstub.py or by WINGDB_ATTACHPORT environment variable before importing wingdbstub in the debug process.

Interner Name:

```
debug.attach-defaults
```

Daten Spezifikation:

```
[tuple von: [tuple Länge 2 von: <type str>, [von 0 bis 65535]]]
```

Standardeinstellung:

```
(('127.0.0.1', 50015),)
```

- **Erweitert**

**Netzwerkabschaltung**

Kontrolliert die Zeitdauer, die der Debug-Client auf eine Reaktion des Debug-Servers wartet, bevor er abbricht. Dies schützt das IDE vor dem Einfrieren, wenn Ihr Programm, das innerhalb des Debug-Servers läuft, abstürzt (oder wenn der Server selbst unerreichbar wird). Es muss auch beachtet werden, wenn die Netzwerkverbindungen langsam sind oder große Datenwerte versendet werden. (Siehe die Einstellungen Große Listenschwelle und Große Stringschwelle).

Interner Name:

```
debug.network-timeout
```

Daten Spezifikation:

```
<type float>, <type int>
```

Standardeinstellung:

```
10
```

### Datenwarnungen anzeigen

Kontrolliert, ob das Abschalten, große Werte und die Fehlerbehandlung der Wertefehler durch den Debugger angezeigt werden, wenn sie das erste Mal in jedem Durchlauf von Wing auftreten.

Interner Name:

```
debug.show-debug-data-warnings
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

### sys.stdin Wrapper verwenden

Prüft, ob sys.stdin als ein Wrapper-Objekt für Nutzereingaben in dem Programm, das debuggt wird, eingestellt werden sollte. Der Wrapper erlaubt, dass Debug-Befehle, wie Anhalten, ausgeführt werden, während das Programm auf Nutzereingaben wartet. Der Wrapper kann Probleme mit Mehrpfadprogammen (Multi-Threaded) verursachen, die C stdio Funktionen nutzen, um direkt vom stdin zu lesen, verursachen und wird langsamer als das normale Dateiobjekt sein. Das Abschalten dieser Einstellung bedeutet jedoch, dass Ihr Debug-Prozess nicht anhält oder Haltepunktänderungen nicht akzeptiert, während auf Tastatureingaben gewartet wird; und jegliche Tastatureingaben, die als Nebeneffekt von Befehlen, die in den Debug-Test eingegeben werden, auftreten, werden stattdessen in ungeändertem erb!stdin! geschehen (obwohl Ausgabe noch wie immer im Debug-Test erscheint.)

Interner Name:

```
debug.use-stdin-wrapper
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

**Protokolldatei der Debug-Internals**

Dies wird genutzt, um ausführliche Informationen über die Debugger- Systemarchitektur zu erhalten, wenn Sie Probleme haben, das Debuggen zum laufen zu bringen. Wenn es auf non-None Wert eingestellt ist, wird die Debugger-Akitivität in dem gegebenen Dateinamen protokolliert. Alternativ können „<stdout>“ oder „<stderr>“ genutzt werden.

Interner Name:

```
debug.logfile
```

Daten Spezifikation:

```
[einer von: None, [<stdout>, <stderr>], <type str>]
```

Standardeinstellung:

```
None
```

**Shells Ignore Editor Modes**

Set to False so that shells will act modal in the same way as editors when working with a modal key bindings such as that for VI. When True, the shells always act as if in Insert mode.

Interner Name:

```
debug.shells-ignore-editor-modes
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

# Source-Analyse

### Im Hintergrund analysieren

Stellt ein, ob Wing versuchen sollte, Python-Source im Hintergrund zu analysieren.

Interner Name:

```
pysource.analyze-in-background
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

### Maximale Cache-Größe (MB)

Die Maximalgröße des Laufwerkspeichers in Megabyte

Interner Name:

```
pysource.max-disk-cache-size
```

Daten Spezifikation:

```
[von 1 bis 1000]
```

Standardeinstellung:

```
50
```

### Maximaler Zwischenspeicher

Die maximale Anzahl # von Analyse-Informations-Puffern, die gleichzeitig im Speicher sein können für Dateien, die nicht geöffnet sind.

Interner Name:

```
pysource.max-background-buffers
```

Daten Spezifikation:

```
[von 1 bis 100]
```

Standardeinstellung:

```
40
```

### Wartezeit für das Eintippen von Buchstabenfolgen

Anzahl der Sekunden zwischen dem letzten Tastendruck und dem Zeitpunkt, an dem die Analyse neu aktiviert wird, falls die Analyse während des Tippens unterbrochen werden soll. Wenn<= 0 wird die Analyse nicht unterbrochen.

Interner Name:

```
edit.suspend-analysis-timeout
```

Daten Spezifikation:

```
<type float>, <type int>
```

Standardeinstellung:

```
3
```

- **Erweitert**

### Dateipfad der Schnittstelle

Pfad zu suchen für Schnittstellen-Dateien für Builtin-Module. Wenn der Verzeichnisname relativ ist, wird er relativ zur Benutzer-Einstellungs Verzeichnis (USER_SETTINGS_DIR) interpretiert

Interner Name:

```
pysource.interfaces-path
```

Daten Spezifikation:

```
[tuple von: <type str>]
```

Standardeinstellung:

```
('pi-files',)
```

### Scrape Extension Modules

Set this to False to disable automatic loading of extension modules and other modules that cannot be statically analysed. These modules are loaded in another process space and 'scraped' to obtain at least some analysis of the module's contents.

Interner Name:

```
pysource.scrape-modules
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
True
```

### Scraping Helper Snippets

This is a dictionary from module name to Python code that should be executed before attempting to load extension modules for scraping. This is needed in some cases such as PyGTK and wxPython because the extension modules are designed to be loaded

only after some configuration magic is performed. For most extension modules, no extra configuration should be needed.

Interner Name:

```
pysource.scrape-config
```

Daten Spezifikation:

```
[dict; keys: <type str>, Werte: <type str>]
```

Standardeinstellung:

```
{'gtk': 'import pygtk\nvers = pygtk._get_available_versions().keys()\nvers.s
cept:\n    pass\n', 'atk': 'import pygtk\nvers = pygtk._get_available_versio
cept:\n    pass\n', 'wxpython': 'pass', 'gobject': 'im-
port pygtk\nvers = pygtk._get_available_versions().keys()\nvers.sort()\nvers
cept:\n    pass\n', 'pango': 'import pygtk\nvers = pygtk._get_available_vers
cept:\n    pass\n', 'gdk': 'import pygtk\nvers = pygtk._get_available_versio
cept:\n    pass\n'}
```

# IDE Extension Scripting

### Pfad suchen

Specifies the directories in which Wing will look for user-defined scripts that extend the functionality of the IDE itself. For each directory, Wing will load all found Python modules and packages, treating any function whose name starts with a letter (not _ or __) as a script-provided command. Extension scripts found in files within directories later in the list will override scripts of the same name found earlier, except that scripts can never override commands that are defined internally in Wing itself (these are documented in the Command Reference in the users manual). See the Scripting and Extending chapter of the manual for more information on writing and using extension scripts. Note that WINGHOME/scripts is always appended to the given path since it contains scripts that ship with Wing.

Interner Name:

```
main.script-path
```

Daten Spezifikation:

```
[list von: <type str>]
```

Standardeinstellung:

```
[u'WINGHOME/scripts']
```

### Auto-Reload Scripts on Save

When enabled, Wing will automatically reload scripts that extend the IDE when they are edited and saved from the IDE. This makes developing extension scripts for the IDE very fast, and should work in most cases. Disable this when working on extension scripts that do not reload properly, such as those that reach through the scripting API extensively.

Interner Name:

```
main.auto-reload-scripts
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
True
```

# Interne Einstellungen

# Haupteinstellungen

### main.debug-break-on-critical

Wenn Wahr und eine kritische gtk-, gdk- oder glib-Nachricht protokolliert wurde, versucht Wing, einen C-Debugger zu starten und am aktuellen Ausführungspunkt zu halten.

Interner Name:

```
main.debug-break-on-critical
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
False
```

**main.extra-mime-type-comments**

Dies ist eine Konvertierung von Mime-Typen zu Tuple von Anfang/End-Kommentarzeichen für jeden Mime-Typen. Ein Zugang sollte für jeden neuen Mime-Typen, der mit der Einstellung main.extra-mime-types hinzugefügt wurde, erstellt werden.

Interner Name:

```
main.extra-mime-type-comments
```

Daten Spezifikation:

```
[dict; keys: <type str>, Werte: [tuple Länge 2 von: <ty-
pe str>, <type str>]]
```

Standardeinstellung:

```
{}
```

**main.extra-mime-type-names**

Dies ist eine Konvertierung von Mime-Typen zu anzeigbaren Namen für diese Mime-Typen; ein Zugang sollte für jeden neuen Mime-Typen, der mit der Einstellung main.extra-mime-types hinzugefügt wurde, erstellt werden.

Interner Name:

```
main.extra-mime-type-names
```

Daten Spezifikation:

```
[dict; keys: <type str>, Werte: <type str>]
```

Standardeinstellung:

```
{}
```

### main.ignored-updates

Used internally to keep track of updates the user is not interested in

Interner Name:

```
main.ignored-updates
```

Daten Spezifikation:

```
[list von: <type str>]
```

Standardeinstellung:

```
[]
```

## Einstellungen der Benutzeroberfläche

### gui.apple-keyboard

Whether an Apple keyboard is in use. Use query x11 option to attempt to determine setting from X11 server each time Wing is run. This is an OS X only preference.

Interner Name:

```
gui.apple-keyboard
```

Daten Spezifikation:

```
[query-x11, yes, no]
```

Standardeinstellung:

```
query-x11
```

**gui.feedback-email**

E-Mail-Adresse, die standardmäßig in den Feedback- und Fehlerbericht-Dialogen verwendet wird

Interner Name:

```
gui.feedback-email
```

Daten Spezifikation:

```
<type str>
```

Standardeinstellung:

```
""
```

**gui.fix-osx-tiger-keyboard-conflict**

Whether to fix the inability to use Mode_switch on Tiger (OS X 10.4). If true, Wing will run xmodmap when it starts to remap the Mode_switch keys (option, Alt Gr, and other composition keys on non-US keyboards) from mod1 to mod5. The xmodmap modifications will affect all X11 applications.

Interner Name:

```
gui.fix-osx-tiger-keyboard-conflict
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
True
```

**gui.osx-key-for-alt**

Use key for alt key in all X11 applications on OS X -- typically used when using a non OS X keyboard layout on the Apple X11 server. This will use xmodmap to set the global X11 key map to use the specified key as the alt key modifier. Turning this option off if it was on previously will reset the option key back to mode_switch, which is the Apple default setting. Non-default options will override any externally set xmodmap setting so use with care if you've customized your xmodmap.

Interner Name:

```
gui.osx-key-for-alt
```

Daten Spezifikation:

```
[default, command, option]
```

Standardeinstellung:

```
default
```

**gui.include-file-types**

Kontrolliert, welche Dateitypen für Mehrdatei-Operationen, wie Suchen und Importieren von Dateien in ein Projekt, berücksichtigt werden.

Interner Name:

```
gui.include-file-types
```

Daten Spezifikation:

```
[tuple von: <type str>]
```

Standardeinstellung:

```
('*.*',)
```

**gui.last-feedback-shown**

Wählt ob das Feedbackdialog am Beenden gezeigt wird.

Interner Name:

```
gui.last-feedback-shown
```

Daten Spezifikation:

```
<type float>
```

Standardeinstellung:

```
0.0
```

**gui.omit-file-types**

Listet die Dateitypen auf, die von Mehrdatei-Operationen, wie Suchen und Importieren von Dateien in ein Projekt, ausgeschlossen werden sollten. Diese werden auch dann ausgeschlossen, wenn die gui.include-file-types Einstellung eine übereinstimmende Wildcard beinhaltet.

Interner Name:

```
gui.omit-file-types
```

Daten Spezifikation:

```
[tuple von: <type str>]
```

Standardeinstellung:

```
('*.o', '*.a', '*.so', '*.pyc', '*.pyo', 'core', '*~', '#*#', 'CVS', '.#*')
```

**gui.prefered-symbol-order**

Kontrolliert die bevorzugte Reihenfolge in Source-Index-Anzeigen, wie den Durchsuchen-Menüs des Editors. Sortiert entweder in „Datei-Reihenfolge“ oder in „Alphabetischer Reihenfolge“.

Interner Name:

```
gui.prefered-symbol-order
```

Daten Spezifikation:

```
[file-order, alpha-order]
```

Standardeinstellung:

```
alpha-order
```

**gui.reported-exceptions**

Intern verwendet, um zu speichern, welche unerwarteten Exceptions bereits berichtet wurden, so dass wir für jeden Fehler nur einen Fehlerbericht-Dialog anzeigen. Dies ist ein dict von Produktversion zu dict der Exception-Info.

Interner Name:

```
gui.reported-exceptions
```

Daten Spezifikation:

```
[dict; keys: <type str>, Werte: [dict; keys: <type str>, Wer-
te: <boolean: 0 oder 1>]]
```

Standardeinstellung:

```
{}
```

**gui.scan-for-pythoncom-shell-extensions**

Suche nach pythoncom shell extensions unter Windows

Interner Name:

```
gui.scan-for-pythoncom-shell-extensions
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
True
```

### gui.set-win32-foreground-lock-timeout

Kontrolliert, ob die Zeitabschaltung der Vordergrundsperre in Windows 98/ME und 2K/XP eingestellt ist. In diesen Systemen ist Wing normalerweise nicht in der Lage, Source-Fenster immer dann in den Vordergrund zu bringen, wenn der Debug-Prozess Fenster im Vordergrund hat. Wenn diese Einstellung wahr ist, wird der systemweite Wert, der Hintergrundanwendungen davon abhält, Fenster aufzuschlagen, immer dann aufgehoben, wenn Wing läuft. Das bedeutet, dass andere Anwendungen auch in der Lage sein werden, Fenster ohne diese Einschränkungen aufzuschlagen, wenn Wing läuft. Setzen Sie die Einstellung auf falsch, um dies zu vermeiden, aber rechnen Sie damit, dass das Aufschlagen von Fenstern in einigen Fällen fehlschlagen kann. Beachten Sie: Wenn Wing fehlerhaft oder vom Taskmanager beendet wird, bleibt der geänderte Wert bestehen, bis sich der Nutzer abmeldet (oder auf 98/ME neu startet).

Interner Name:

```
gui.set-win32-foreground-lock-timeout
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

### gui.show-feedback-dialog

Bestimmt, ob der Feedback-Dialog dem Benutzer beim Beenden angezeigt wird.

Interner Name:

```
gui.show-feedback-dialog
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

### gui.show-osx-keyboard-warning

Used internally to show information about osx keyboard issues to new users. Once turned off, it is never turned on again

Interner Name:

```
gui.show-osx-keyboard-warning
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

### gui.startup-show-wingtips

Kontolliert, ob das Wing Tipps-Werkzeug automatisch beim Start des IDE angezeigt wird.

Interner Name:

```
gui.startup-show-wingtips
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

## Einstellungen des Editors

### edit.fold-mime-types

Auf eine Liste von Mime-Typen einstellen, für welche das Falten erlaubt sein sollte, wenn Falten im Allgemeinen aktiviert ist.

Interner Name:

```
edit.fold-mime-types
```

Daten Spezifikation:

```
[list von: <type str>]
```

Standardeinstellung:

```
['text/x-python', 'text/x-c-source', 'text/x-cpp-source', 'text/x-
java-source', 'text/x-javascript', 'text/html', 'text/xml', 'text/x-
eiffel', 'text/x-lisp', 'text/x-ruby']
```

**edit.use-default-foreground-when-printing**

Use default foreground color for all text when printing. It's to set this if foreground color are customized for display on a dark background. The background color when printing is assumed to be white.

Interner Name:

```
edit.use-default-foreground-when-printing
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
False
```

# Einstellungen des Projektmanagers

**proj.follow-selection**

Kontrolliert, ob das IDE der aktuellen Projektmanager-Auswahl folgen wird, indem es die entsprechende Source-Datei in einem nicht-sticky (automatisch schließenden) Editor öffnet. In jedem Fall wird der Projektmanager eine Datei immer im Sticky-Modus öffnen, wenn ein Eintrag doppelt angeklickt wird oder der Menüeintrag „Gehe zur Source“ verwendet wird.

Interner Name:

```
proj.follow-selection
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

## Einstellungen des Debuggers

### debug.auto-clear-debug-io

Stellt ein, dass der Text des Debug-I/O jedesmal, wenn eine neue Debug-Sitzung gestartet wird, automatisch gelöscht wird.

Interner Name:

```
debug.auto-clear-debug-io
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

### debug.python-exec

Stellen Sie dies ein, um die voreingestellte Python-Executable, die mit dem Debug-Server genutzt wird, außer Kraft zu setzen. Ein None (Voreinstellung) Wert nutzt /usr/bin/env Python auf Linux und der konfigurierten Voreinstellung auf NT. Andernfalls geben Sie den gesamten Pfad der Python-Executable an, z.B. /usr/local/bin/python oder C:devpython. Diese Einstellung wirkt sich nur auf Programme aus, die vom IDE aus gestartet werden.

Interner Name:

```
debug.python-exec
```

Daten Spezifikation:

```
[None oder <type str>]
```

Standardeinstellung:

```
None
```

**debug.safe-size-checks-only**

Dies ist eine *temporäre* Einstellung, die in der zukünftigen Version von Wing IDE verschwinden wird. Sie kann genutzt werden, um Server-seitige Größenkontrollen von Werten, die in der interaktiven Shell eingegeben wurden, auszuschalten. Wenn es auf wahr eingestellt wird, kann Wing den Debug-Prozess für große Werte, die in der interaktiven Shell bewertet werden, beenden. Wenn es auf falsch einestellt ist, wird Wing Größenüberprüfungen durchführen, um eine solches Beenden zu vermeiden, aber es wird auch eine doppelte Ausführung von jeder Funktionalität, die als Ergebnis von __getattr__ method erreicht wird, verursachen.

Interner Name:

```
debug.safe-size-checks-only
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

**debug.show-exceptions-tip**

Wird intern verwendet, um Informationen über die Exception-Behandlung für neue Nutzer anzuzeigen. Wenn es einmal ausgeschalten wird, dann wird es nie wieder angeschalten

Interner Name:

```
debug.show-exceptions-tip
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

**debug.stop-timeout**

Anzahl der zu wartenden Sekunden, bevor der Debugger in seinem eigenen Code anhält, nachdem eine Anhalteanfrage empfangen wurde und kein anderer Python-Code erreicht wird.

Interner Name:

```
debug.stop-timeout
```

Daten Spezifikation:

```
<type int>, <type float>
```

Standardeinstellung:

```
3.0
```

**debug.use-members-attrib**

Auf wahr einstellen, damit der Debug-Server das __members__ Attribut nutzt, um andernfalls zu versuchen, unlesbare Datenwerte zu interpretieren. Dies ist eine Grundeinstellung, da einige Erweiterungsmodule Fehler enthalten, die Programmabstürze verursachen, wenn auf dieses Atttribut zugegriffen wird. Beachten Sie, dass __members__ seit der Python-Version 2.2 nicht ausgeführt wurde.

Interner Name:

```
debug.use-members-attrib
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
1
```

**debug.wrap-debug-io**

Auf wahr setzen, um Zeilenumbruch in dem integrierten Debug-I/O-Feld anzuschalten.

Interner Name:

```
debug.wrap-debug-io
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

**debug.wrap-debug-probe**

Auf wahr setzen, um Zeilenumbruch in dem integrierten Debug-Test-Feld anzuschalten.

Interner Name:

```
debug.wrap-debug-probe
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

**debug.wrap-python-shell**

Auf wahr setzen, um Zeilenumbruch in dem Python-Shell-Feld anzuschalten.

Interner Name:

```
debug.wrap-python-shell
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

# Einstellungen der Source-Analyse

### pysource.instance-attrib-scan-mode

Wie nach Instanz-Attributen gesucht werden soll.

Interner Name:

```
pysource.instance-attrib-scan-mode
```

Daten Spezifikation:

```
[init-only, all-methods]
```

Standardeinstellung:

```
all-methods
```

# Einstellungen des Source-Browsers

### browser.follow-selection

Kontrolliert, ob das IDE der aktuellen Browser-Auswahl folgen wird, indem es die entsprechende Source-Datei in einem nicht-sticky (automatisch schließenden) Editor öffnet. In jedem Fall wird der Browser eine Datei immer im Sticky-Modus öffnen, wenn ein Eintrag doppelt angeklickt wird oder der Menüeintrag „Gehe zur Source“ verwendet wird.

Interner Name:

```
browser.follow-selection
```

Daten Spezifikation:

```
<boolean: 0 oder 1>
```

Standardeinstellung:

```
0
```

# Befehlsreferenz

Dieses Kapitel umfasst das gesamte Top-Level Befehlsset von Wing IDE. Verwenden Sie diese Referenz, um Befehlsnamen für die Verwendung in geänderten **Tastaturkombinationen** nachzuschlagen.

Explicit markup ends without a blank line; unexpected unindent.

## Top Level Commands

Dies sind die Top-Level Anwendungsbefehle.

**abandon-changes** (confirm=True)

Abandon any changes in the current document and reload it from disk. Prompts for user to confirm the operation unless either there are no local changes being abandoned or confirm is set to False.

**about-application** ()

Die anwendungsweite Über-Box anzeigen

**begin-visited-document-cycle** (move_back=True)

Start moving between documents in the order they were visited. Starts modal key interation that ends when a key other than tab is seen or ctrl is released.

**check-for-updates** ()

Check for updates to Wing IDE and offer to install any that are available

**close** (ignore_changes=False, close_window=False)

Close active document. Abandon any changes when ignore_changes is True. Close empty windows and quit if all document windows closed when close_window is True.

**close-all** (omit_current=False, ignore_changes=False, close_window=False)

Close all documents in the current window, or in all windows if in one-window-per-editor windowing policy. Leave currently visible documents (or active window in one-window-per-editor-mode) if omit_current is True. Abandons changes rather than saving them when ignore_changes is True. Close empty window and quit if all document windows closed when close_window is True.

**close-window** ()

Das aktuelle Fenster und alle Dokumente und Felder in diesem schließen

**command-by-name** (command_name)

Gegebenen Befehl nach Namen ausführen, irgendwelche Argumente sammeln, wie benötigt

**copy-tutorial** ()

Kopiert das Tutorial-Verzeichnis aus der Wing IDE Installation in ein Verzeichnis, das vom Benutzer gewahlt wird.

**edit-file-sets** ()

Editor für die Einstellungen der Dateisets anzeigen

**edit-preferences-file** ()

Die Einstellungen als eine Textdatei bearbeiten

**execute-cmd** (cmd)

Execute the given command line silently in the background

**execute-file** (loc=None)

Die Datei am gegebenen Ort ausführen oder die aktive Ansicht verwenden, wenn loc None ist.

**goto-bookmark** (mark)

Goto named bookmark

**initiate-numeric-modifier** (digit)

VI style repeat/numeric modifier for following command

**initiate-repeat** ()

Eine Reihenfolge von Ziffern eingeben, die die Anzahl der Wiederholungen des nachfolgenden Befehls oder Tastenanschlags anzeigt.

**initiate-repeat-0** ()

Eine Reihenfolge von Ziffern eingeben, die die Anzahl der Wiederholungen des nachfolgenden Befehls oder Tastenanschlags anzeigt.

**initiate-repeat-1** ()

Eine Reihenfolge von Ziffern eingeben, die die Anzahl der Wiederholungen des nachfolgenden Befehls oder Tastenanschlags anzeigt.

**initiate-repeat-2** ()

Eine Reihenfolge von Ziffern eingeben, die die Anzahl der Wiederholungen des nachfolgenden Befehls oder Tastenanschlags anzeigt.

**initiate-repeat-3** ()

Eine Reihenfolge von Ziffern eingeben, die die Anzahl der Wiederholungen des nachfolgenden Befehls oder Tastenanschlags anzeigt.

**initiate-repeat-4** ()

Eine Reihenfolge von Ziffern eingeben, die die Anzahl der Wiederholungen des nachfolgenden Befehls oder Tastenanschlags anzeigt.

**initiate-repeat-5** ()

Eine Reihenfolge von Ziffern eingeben, die die Anzahl der Wiederholungen des nachfolgenden Befehls oder Tastenanschlags anzeigt.

**initiate-repeat-6** ()

Eine Reihenfolge von Ziffern eingeben, die die Anzahl der Wiederholungen des nachfolgenden Befehls oder Tastenanschlags anzeigt.

**initiate-repeat-7** ()

Eine Reihenfolge von Ziffern eingeben, die die Anzahl der Wiederholungen des nachfolgenden Befehls oder Tastenanschlags anzeigt.

**initiate-repeat-8** ()

Eine Reihenfolge von Ziffern eingeben, die die Anzahl der Wiederholungen des nachfolgenden Befehls oder Tastenanschlags anzeigt.

**initiate-repeat-9** ()

Eine Reihenfolge von Ziffern eingeben, die die Anzahl der Wiederholungen des nachfolgenden Befehls oder Tastenanschlags anzeigt.

**internal-profile-start** ()

Start internal profiling.

**internal-profile-stop** ()

Stop internal profiling.

**new-blank-file** (filename)

Create a new blank file on disk, open it in an editor, and add it to the current project.

**new-document-window** ()

Ein neues Dokumentfenster mit den gleichen Dokumenten und Feldern wie im aktuellen Dokumentfenster (falls vorhanden, ansonsten leer mit Standard-Feldern) erstellen

**new-file** (ext='.py')

Eine neue Datei erstellen

**new-panel-window** (panel_type=None)

Ein neues Feldfenster des gegebenen Typs erstellen

**next-document** ()

In der Liste der Dokumente, die im aktuellen Fenster geöffnet sind, alphabetisch zum nächsten Dokument gehen

**next-window** ()

Zum nächsten Fenster in Alphabetischer Reihenfolge

**open** (filename)

Eine Datei vom Laufwerk öffnen

**open-gui** (filename=None)

Eine Datei vom Laufwerk öffnen, Aufforderung mit Dialog Dateiauswahl, wenn erforderlich

**previous-document** ()

In der Liste der Dokumente, die im aktuellen Fenster geöffnet sind, alphabetisch zum vorherigen Dokument gehen

**previous-window** ()

None

**query-end-session** ()

Query-end-session Nachricht auf win32 verarbeiten

**quit** ()

Anwendung beenden.

**recent-document** ()

Wechselt zum vorherigen Dokument, das zuletzt im aktuellen Fenster oder im Fensterset, wenn Sie sich im Fenstermodus Ein-Fenster-pro-Editor befinden, besucht wurde.

**reload-scripts** ()

Alle Skripte neu laden, von alle eingestellten Skriptverzeichnisse. Dieses is meistens nur nötig wenn ein neues Skript Modul zu gefügt wird. Existierende Skript Module werden automatisch neu geladen wenn sie am Laufwerk neu gespeichert werden.

**remove-bookmark** (mark)

Remove the given named bookmark

**restore-default-tools** ()

Alle Werkzeuge verstecken/entfernen und den ursprünglichen Standard-Zustand wiederherstellen

**save** (close=False, force=False)

Save active document. Also close it if close is True.

**save-all** (close_window=False)

Alle ungespeicherten Objekte speichern. Wird den Nutzer nur auffordern, für neue Objekte, die keinen Dateinamen haben, einen Namen zu wählen

**save-as** ()

Aktives Dokument in einer neuen Datei speichern

**scratch-document** (title='Scratch', mime_type='text/plain')

Ein neues Notizspeicher aufschlagen mit bestimmten title und Mime-Typ. Der Puffer wird nie als beendert markiert aber kann unter einen anderen Namen gespeichert werden.

**set-bookmark** (mark)

Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**show-bookmarks** ()

Show a list of all currently defined bookmarks

**show-bug-report-dialog** ()

Dialog für Fehlerberichte anzeigen

**show-document** (section='manual')

Gegebenen Dokumentationsabschnitt anzeigen

**show-feedback-dialog** ()

Dialog für Feedback anzeigen

**show-howtos** ()

How-Tos-Index anzeigen

**show-html-document** (section='manual')

Show the given document section in HTML format.

**show-manual-html** ()

HTML-Version des Wing IDE Benutzerhandbuches anzeigen

**show-manual-pdf** ()

PDF-Version des Wing IDE Benutzerhandbuches anzeigen, entweder in US Letter oder A4, abhängig vom Druckort des Nutzers

**show-panel** (panel_type, flash=True)

Zeige letztes Instanz von einen Tool. Wenn keine Instanze existieren, füge eines den Haubtfenster zu. Sendet zurück das Instanz oder None nach ein Fehler.

**show-pdf-document** (doc='manual')

Show the given document in PDF format. One of 'manual', 'intro', or 'howtos'.

**show-preferences-gui** (prefname=None)

Die Einstellungsdatei, die die GUI-Einstellungen nutzt, bearbeiten; optional im Abschnitt öffnen, der die gegebenen Einstellungen nach Namen enthält

**show-python-for-beginners-html** ()

Die Webseite Python für Anfänger anzeigen

**show-python-introductions-html** ()

Die Webseite Python Einführungen anzeigen

**show-python-manual-html** ()

HTML-Version des Python-Benutzerhandbuches anzeigen

**show-python-org-html** ()

Homepage der python.org Site anzeigen

**show-python-org-search-html** ()

Suchseite der python.org Seite anzeigen

**show-quickstart** ()

Schnellstart-Anleitung anzeigen

**show-success-stories-html** ()

Seite der Python Success Stories anzeigen

**show-support-html** ()

Wing IDE Support Site Homepage anzeigen

**show-text-registers** ()

Show the contents of all non-empty text registers in a temporary editor

**show-tutorial** ()

Tutorial anzeigen

**show-wingtip** (section='/')

Fenster mit Wing Tipps anzeigen

**switch-document** (document_name)

Wechselt zum genannten Dokument. Name kann entweder ein vollständiger Name oder die letzte Pfadkomponente eines Pfadnamens sein.

**toolbar-search** (text, next=False, set_anchor=True)

Von der aktuellen Cursor- oder Auswahlposition suchen, und zwar unter Verwendung des Textes, der im Suchbereich der Werkzeugleiste eingegeben ist.

**toolbar-search-next** (text, set_anchor=True)

Zum nächsten Treffer des Texts, der im Suchbereich der Werkzeugleiste eingegeben ist, gehen

**vi-goto-bookmark** ()

Goto bookmark using single character name defined by the next pressed key

**vi-set-bookmark** ()

Set a bookmark at current location on the editor using the next key press as the name of the bookmark.

**wing-tips** ()

Interaktiven Hinweismanager anzeigen

**write-changed-file-and-close** (filename)

Write current document to given location only if it contains any changes and close it. Writes to current file name if given filename is None.

**write-file** (filename)

Write current file to a new location.

**write-file-and-close** (filename)

Write current document to given location and close it. Saves to current file name if the given filename is None.

# Dock Window Commands

Befehle für Fenster, die ankoppelbare Werkzeugbereiche enthalten. Diese sind für das derzeitig aktive Fenster, wenn vorhanden, verfügbar.

**enter-fullscreen** ()

Hide both the vertical and horizontal tool areas and toolbar, saving previous state so it can be restored later with exit_fullscreen

**exit-fullscreen** ()

Restore previous non-fullscreen state of all tools and tool bar

**hide-horizontal-tools** ()

Hide the horizontal tool area

**hide-vertical-tools** ()

Hide the vertical tool area

**minimize-horizontal-tools** ()

Waagerechten Werkzeugbereich minimieren

**minimize-vertical-tools** ()

Senkrechten Werkzeugbereich minimieren

**show-horizontal-tools** ()

Waagerechten Werkzeugbereich anzeigen

**show-vertical-tools** ()

Senkrechten Werkzeugbereich anzeigen

**toggle-horizontal-tools** ()

Den waagerechten Werkzeugbereich anzeigen oder minimieren

**toggle-vertical-tools** ()

Den senkrechten Werkzeugbereich anzeigen oder minimieren

# Document Viewer Commands

Befehle für die Dokumentationsansicht. Diese stehen zur Verfügung, wenn die Tastatur für die Dokumentationsansicht aktiviert ist.

**document-back** ()

Zurückgehen zur vorherigen Dokumentseite in der Historie der angesehenen Seiten

**document-contents** ()

Zur Inhaltsseite des Dokuments gehen

**document-forward** ()

Vorwärtsgehen zur nächsten Dokumentseite in der Historie der angesehenen Seiten

**document-next** ()

Zur nächsten Seite im aktuellen Dokument gehen

**document-previous** ()

Zur vorherigen Seite im aktuellen Dokument gehen

**isearch-backward** (search_string=None, repeat=<command.commandmgr.kArgNumericModifie
instance at 0x4144b98c>)

Initiate incremental mini-search backward from the cursor position, optionally entering the given search string.

**isearch-backward-regex** (search_string=None, repeat=<command.commandmgr.kArgNumeric
instance at 0x4144b9ac>)

Initiate incremental regular expression mini-search backward from the cursor position, optionally entering the given search string.

**isearch-forward** (search_string=None, repeat=<command.commandmgr.kArgNumericModifier
instance at 0x4144b92c>)

Initiate incremental mini-search forward from the cursor position, optionally entering the given search string.

**isearch-forward-regex** (search_string=None, repeat=<command.commandmgr.kArgNumericM
instance at 0x4144b96c>)

Initiate incremental regular expression mini-search forward from the cursor position, optionally entering the given search string.

**isearch-repeat** (reverse=False, repeat=<command.commandmgr.kArgNumericModifier
instance at 0x4144b9cc>)

Repeat the most recent isearch, using same string and regex/text. Reverse direction when reverse is True.

**isearch-sel-backward** (persist=True, repeat=<command.commandmgr.kArgNumericModifier
instance at 0x4144ba4c>)

Initiate incremental mini-search backward from the cursor position, using current selection as the search string. Set persist=False to do the search but end the interactive search session immediately.

**isearch-sel-forward** (persist=True, repeat=<command.commandmgr.kArgNumericModifier instance at 0x4144ba0c>)

Initiate incremental mini-search forward from the cursor position, using current selection as the search string. Set persist=False to do the search but end the interactive search session immediately.

**repeat-search-char** (opposite=0, repeat=<command.commandmgr.kArgNumericModifier instance at 0x4144ba8c>)

Repeat the last search_char operation, optionally in the opposite direction.

**search-char** (dir=1, pos=0, repeat=<command.commandmgr.kArgNumericModifier instance at 0x4144ba6c>, single_line=0)

Search for the given character. Searches to right if dir > 0 and to left if dir < 0. Optionally place cursor pos characters to left or right of the target (e.g., use -1 to place one to left). If repeat > 1, the Nth match is found. Set single_line=1 to search only within the current line.

# Editor Browse Mode Commands

Commands available only when the editor is in browse mode (used for VI bindings and possibly others)

**enter-insert-mode** (pos='before')

Enter editor insert mode

**enter-replace-mode** ()

Enter editor replace mode

**enter-visual-mode** (unit='char')

Enter editor visual mode. Unit should be one of 'char', 'line', or 'block'.

**start-select-block** ()

Turn on auto-select block mode

**start-select-char** ()

Turn on auto-select mode character by character

**start-select-line** ()

Turn on auto-select mode line by line

**vi-command-by-name** ()

Execute a VI command (implements „:“ commands from VI)

**vi-ctrl-c** ()

Perform vi mode ctrl-c action which either does a copy or nothing if ctrl-x/v/c are not being used for clipboard actions. The default is to map ctrl-c to clipboard on Windows and OS X. This can be overridden by the VI Mode Ctrl-X/C/V preference.

**vi-ctrl-v** ()

Perform vi mode ctrl-v action which either does a paste or does start-select-block. The default is to map ctrl-v to clipboard on Windows and OS X. This can be overridden by the VI Mode Ctrl-X/C/V preference.

**vi-ctrl-x** ()

Perform vi mode ctrl-x action which either does a cut or does initiate-numeric-modified with the following digit key press. The default is to map ctrl-x to clipboard on Windows and OS X. This can be overridden by the VI Mode Ctrl-X/C/V preference.

# Editor Insert Mode Commands

Commands available only when editor is in insert mode (used for VI bindings and possibly others)

**enter-browse-mode** (provisional=False)

Enter editor browse mode

**vi-ctrl-c** ()

Perform vi mode ctrl-c action which either does a copy or enters browse mode if ctrl-x/v/c are not being used for clipboard actions. The default is to map ctrl-c to clipboard on Windows and OS X. This can be overridden by the VI Mode Ctrl-X/C/V preference.

**vi-ctrl-v** ()

Perform vi mode ctrl-v action which either does a paste or does start-select-block. The

default is to map ctrl-v to clipboard on Windows and OS X. This can be overridden by the VI Mode Ctrl-X/C/V preference.

**vi-ctrl-x** ()

Perform vi mode ctrl-x action which either does a cut or nothing depending on whether ctrl-x/v/c are mapped to clipboard actions. The default is to map ctrl-x to clipboard on Windows and OS X. This can be overridden by the VI Mode Ctrl-X/C/V preference.

# Editor Non Modal Commands

Commands available only when the editor is in non-modal editing mode

**start-select-block** ()

Turn on auto-select block mode

**start-select-char** ()

Turn on auto-select mode character by character

**start-select-line** ()

Turn on auto-select mode line by line

# Editor Panel Commands

Befehle, die das Teilen eines Editor-Feldes steuern. Diese sind verfügbar, wenn die Tastatur für einen Teil im Editor-Feld aktiviert ist.

**split-horizontally** (new=0)

Aktuelle Ansicht waagerecht teilen.

**split-vertically** (new=0)

Split current view vertically. Create new editor in new view when new==1.

**split-vertically-open-file** (filename)

Split current view vertically and open selected file

**unsplit** (action='current')

Unsplit all editors so there's only one. Action specifies how to choose the remaining displayed editor. One of:

current -- Show current editor close -- Close current editor before unsplitting recent -- Change to recent buffer before unsplitting recent-or-close -- Change to recent buffer before closing

Unexpected indentation.

> split, or close the current buffer if there is only one split left.

NOTE: The parameters for this command are subject to change in the future.

# Editor Replace Mode Commands

Commands available only when editor is in replace mode (used for VI bindings and possibly others)

**enter-browse-mode** (provisional=False)

Enter editor browse mode

# Editor Split Commands

Befehle für einen bestimmten Editor-Teil. Diese sind nur verfügbar, wenn die Tastatur für den Editor-Teil aktiviert ist. Zusätzliche Befehle, die den Inhalt des Editors beeinflussen sind separat definiert.

**activate-file-option-menu** ()

Dateimenü für den Editor aktivieren.

**grow-split-horizontally** ()

Increase width of this split

**grow-split-vertically** ()

Increase height of this split

**next-bookmark** ()

Vorwärts zu das nächste Auto-Lesezeichen im Editor

**previous-bookmark** ()

Zurück zu das letzte Auto-Lesezeichen im Editor

**shrink-split-horizontally** ()

Decrease width of this split

**shrink-split-vertically** ()

Decrease height of this split

## Editor Visual Mode Commands

Commands available only when the editor is in visual mode (used for VI bindings and some others)

**enter-browse-mode** ()

Enter editor browse mode

**enter-insert-mode** (pos='delete-sel')

Enter editor insert mode

**enter-visual-mode** (unit='char')

Alter type of editor visual mode or exit back to browse mode. Unit should be one of 'char', 'line', or 'block'.

**exit-visual-mode** ()

Exit visual mode and return back to default mode

**vi-command-by-name** ()

Execute a VI command (implements „:“ commands from VI)

## Global Documentation Commands

Befehle für die Dokumentationsansicht. Diese stehen zur Verfügung, wenn die Tastatur für die Dokumentationsansicht aktiviert ist.

**document-search** (txt=None)

Search all documentation.

# Toolbar Search Commands

Befehle für das Suchfeld in der Werkzeugleiste. Diese stehen zur Verfügung, wenn die Tastatur für das Suchfeld in der Werkzeugleiste aktiviert ist.

**backward-char** ()

Zurück ein Zeichen

**backward-char-extend** ()

Zurück ein Zeichen, Text-Auswahl an der neue Position anpassen

**backward-delete-char** ()

Ein Zeichen hinter den Cursor löschen

**backward-delete-word** ()

Ein Wort hinter den Cursor löschen

**backward-word** ()

Zurück ein Wort

**backward-word-extend** ()

Zurück ein Wort, Text-Auswahl an der neue Position anpassen

**beginning-of-line** ()

Zurück zum Anfang des Suchtextes in der Werkzeugleiste

**beginning-of-line-extend** ()

Zurück zum Anfang des Suchtextes in der Werkzeugleiste, Text-Auswahl an der neue Position anpassen

**copy** ()

Ausgewählten Text ausschneiden

**cut** ()

Ausgewählten Text ausschneiden

**end-of-line** ()

Zum Ende des Suchtextes in der Werkzeugleiste

**end-of-line-extend** ()

Zum Ende des Suchtextes in der Werkzeugleiste, Text-Auswahl an der neue Position anpassen

**forward-char** ()

Vorwärts ein Zeichen

**forward-char-extend** ()

Vorwärts ein Zeichen, Text-Auswahl an der neue Position anpassen

**forward-delete-char** ()

Ein Zeichen vor den Cursor löschen

**forward-delete-word** ()

Ein Wort vor den Cursor löschen

**forward-word** ()

Vorwärts ein Wort

**forward-word-extend** ()

Vorwärts ein Wort, Text-Auswahl an der neue Position anpassen

**paste** ()

Text aus der Zwischenablage einfügen

# Window Commands

Befehle für Fenster. Diese sind für das derzeitig aktive Fenster, wenn vorhanden, verfügbar.

**move-editor-focus** (dir=1, wrap=True)

Move focus to next or previous editor split, optionally wrapping when the end is reached.

**move-editor-focus-first** ()

Move focus to first editor split

**move-editor-focus-last** ()

Move focus to last editor split

**move-editor-focus-previous** ()

Move focus to last editor split

**move-focus** ()

None

# Wing Tips Commands

Befehle für das Werkzeug Wing Tipps. Diese sind nur verfügbar, wenn das Werkzeug sichtbar und aktiviert ist.

**wingtips-close** ()

Das Wing Tipps Fenster schließen

**wingtips-contents** ()

Zur Inhaltsseite der Wing Tipps gehen

**wingtips-next** ()

Zur nächsten Seite der Wing Tipps gehen

**wingtips-next-unseen** ()

Zu die nächste ungesehene Seite der Wing Tipps gehen

**wingtips-previous** ()

Zur vorherigen Seite der Wing Tipps gehen

# Active Editor Commands

Befehle, die nur auf Editoren angewendet werden, für die die Tastatur aktiviert ist. Diese Befehle sind auch für die Werkzeuge Python-Shell, Debug-Test und Debug-I/O verfüg-

bar, die Unterklassen zum Source-Editor bilden, obwohl einige der Befehle in diesem Kontext, wie jeweils erforderlich, verändert oder deaktiviert sind.

**activate-symbol-option-menu-1** ()

Das 1. Symbol-Menü für den Editor aktivieren.

**activate-symbol-option-menu-2** ()

Das 2. Symbol-Menü für den Editor aktivieren.

**activate-symbol-option-menu-3** ()

Das 3. Symbol-Menü für den Editor aktivieren.

**activate-symbol-option-menu-4** ()

Das 4. Symbol-Menü für den Editor aktivieren.

**backward-char** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac52c>)

Cursor ein Zeichen zurück bewegen

**backward-char-extend** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac56c>)

Cursor ein Zeichen zurück bewegen und den Auswahlbereich an die neue Position anpassen

**backward-char-extend-rect** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac5ac>)

Move cursor backward one character, adjusting the rectangular selection range to new position

**backward-delete-char** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac9ac>)

Ein Zeichen nach dem Cursor löschen oder die aktuelle Auswahl wenn nicht leer.

**backward-delete-word** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac9ec>)

Ein Wort hinter dem Cursor löschen

**backward-page** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac7ec>)

Cursor eine Seite zurück bewegen

**backward-page-extend** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac82c>)

Cursor eine Seite zurück bewegen und den Auswahllbereich an die neue Position anpassen

**backward-paragraph** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac6ec>)

Move cursor backward one paragraph (to next all-whitespace line).

**backward-paragraph-extend** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac72c>)

Move cursor backward one paragraph (to next all-whitespace line), adjusting the selection range to new position.

**backward-tab** ()

Zeile an der aktuellen Position ausrücken

**backward-word** (delimiters=None, gravity='start', repeat=<command.commandmgr.kArgNum
instance at 0x412ac62c>)

Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be „start" or „end" to indicate whether cursor is placed at start or end of the word.

**backward-word-extend** (delimiters=None, gravity='start', repeat=<command.commandmgr.k
instance at 0x412ac64c>)

Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be „start" or „end" to indicate whether cursor is placed at start or end of the word.

**beginning-of-line** ()

Gehe zum Start der aktuellen Zeile, oder zum end des Leerraum am Anfang wenn schon am Start der Zeile.

**beginning-of-line-extend** ()

Gehe zum Anfang der aktuellen Zeile, oder zum ende vom Leerraum am Anfang der Zeile wenn schon am Anfang der Zeile, und verschiebe den Auswahlbereich an die neue Position.

**beginning-of-line-text** ()

Gehe zum Ende vom Leerraum am Anfang der aktuelle Zeile oder zum Start der Zeile wenn schon am Ende vom Leerraum.

**beginning-of-line-text-extend** ()

Gehe zum Ende der vom Leerraum am Anfang der aktuellen Zeile, oder zum Start der Zeile wenn schon am Ende vom Leerraum, und verschiebe den Auswahlbereich an die neue Position.

**beginning-of-screen-line** ()

Move to beginning of current wrapped line

**beginning-of-screen-line-extend** ()

Move to beginning of current wrapped line, extending selection

**beginning-of-screen-line-text** ()

Move to first non-blank character at beginning of current wrapped line

**beginning-of-screen-line-text-extend** ()

Move to first non-blank character at beginning of current wrapped line, extending selection

**brace-match** ()

Klammer an der aktuellen Cursor-Position anpassen, gesamten Text zwischen den beiden auswählen und die Klammern markieren

**cancel** ()

Aktuellen Editorbefehl abbrechen

**cancel-autocompletion** ()

Jede aktive Auto-Vervollständigung abbrechen.

**case-lower** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412acc6c>)

Change case of the current selection, or character ahead of the cursor if there is no selection, to lower case

**case-lower-next-move** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412acd6c>)

Change case of text spanned by next cursor movement to lower case

**case-swap** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412accec>)

Change case of the current selection, or character ahead of the cursor if there is no selection, so each letter is the opposite of its current case

**case-swap-next-move** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412acdec>)

Change case of text spanned by next cursor movement so each letter is the opposite of its current case

**case-title** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412accac>)

Change case of the current selection, or character ahead of the cursor if there is no selection, to title case (first letter of each word capitalized)

**case-title-next-move** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412acdac>)

Change case of text spanned by next cursor movement to title case (first letter of each word capitalized)

**case-upper** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412acc0c>)

Change case of the current selection, or character ahead of the cursor if there is no selection, to upper case

**case-upper-next-move** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412acd2c>)

Change case of text spanned by next cursor movement to upper case

**center-cursor** ()

Bild so rollen, dass der Cursor in der Anzeige zentriert wird

**clear** ()

Markierten Text löschen

**complete-autocompletion** ()

Die gegenwärtig aktive Auto-Vervollständigung vervollständigen.

**copy** ()

Markierten Text kopieren

**cursor-move-to-bottom** (offset=<command.commandmgr.kArgNumericModifier instance at 0x412acfec>)

Move cursor to bottom of display (without scrolling), optionally at an offset of given number of lines before bottom

**cursor-move-to-center** ()

Move cursor to center of display (without scrolling)

**cursor-move-to-top** (offset=<command.commandmgr.kArgNumericModifier instance at 0x412acfac>)

Move cursor to top of display (without scrolling), optionally at an offset of given number of lines below top

**cursor-to-bottom** ()

Scroll so cursor is centered at bottom of display

**cursor-to-top** ()

Scroll so cursor is centered at top of display

**cut** ()

Markierten Text ausschneiden

**cut-line** ()

Cut the current line(s) to clipboard.

**delete-line** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412aca2c>)

Delete the current line or lines when the selection spans multiple lines or given repeat is > 1

**delete-line-insert** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412aca6c>)

Delete the current line or lines when the selection spans multiple lines or given repeat is > 1. Enters insert mode (when working with modal key bindings).

**delete-next-move** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412acb4c>)

Delete the text covered by the next cursor move command.

**delete-next-move-insert** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412acb8c>)

Delete the text covered by the next cursor move command and then enter insert mode (when working in a modal editor key binding)

**delete-range** (start_line, end_line, register=None)

Delete given range of lines, copying them into given register (or currently selected default register if register is None

**delete-to-end-of-line** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412acaac>)

Delete everything between the cursor and end of line

**delete-to-end-of-line-insert** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412acaec>)

Delete everything between the cursor and end of line and enter insert move (when working in a modal editor key binding)

**delete-to-start-of-line** ()

Delete everything between the cursor and start of line

**end-of-document** ()

Cursor zum Ende des Dokuments bewegen

**end-of-document-extend** ()

Cursor zum Ende des Dokuments bewegen und den Auswahllbereich an die neue Position anpassen

**end-of-line** (count=<command.commandmgr.kArgNumericModifier instance at 0x412ac22c>)

Move to end of current line

**end-of-line-extend** (count=<command.commandmgr.kArgNumericModifier instance at 0x412ac24c>)

Zum Ende der aktuellen Zeile verschieben und den Auswahlbereich an die neue Position anpassen

**end-of-screen-line** (count=<command.commandmgr.kArgNumericModifier instance at 0x412ac26c>)

Move to end of current wrapped line

**end-of-screen-line-extend** (count=<command.commandmgr.kArgNumericModifier instance at 0x412ac28c>)

Move to end of current wrapped line, extending selection

**exchange-point-and-mark** ()

Wenn Text markiert wird, verwechselt dieses die Startposition und Anker von der Auswahl.

**filter-next-move** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412acbcc>)

Filter the lines covered by the next cursor move command through an external command and replace the lines with the result

**filter-range** (cmd, start_line=0, end_line=-1)

Filter a range of lines in the editor through an external command and replace the lines with the result. Filters the whole file by default.

**filter-selection** (cmd)

Filter the current selection through an external command and replace the lines with the result

**form-feed** ()

Ein Seitenvorschub-Zeichen an der aktuellen Cursor-Position setzen

**forward-char** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac48c>)

Cursor ein Zeichen vorwärts bewegen

**forward-char-extend** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac4cc>)

Cursor ein Zeichen vorwärts bewegen und den Auswahlbereich an die neue Position anpassen

**forward-char-extend-rect** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac4ec>)

Move cursor forward one character, adjusting the rectangular selection range to new position

**forward-delete-char** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac8ac>)

Ein Zeichen vor dem Cursor löschen

**forward-delete-char-insert** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac8ec>)

Delete one char in front of the cursor and enter insert mode (when working in modal key bindings)

**forward-delete-word** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac92c>)

Ein Wort vor dem Cursor löschen

**forward-delete-word-insert** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac96c>)

Delete one word in front of the cursor and enter insert mode (when working in modal key bindings)

**forward-page** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac76c>)

Cursor eine Seite vorwärts bewegen

**forward-page-extend** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac7ac>)

Cursor eine Seite vorwärts bewegen und den Auswahllbereich an die neue Position anpassen

**forward-paragraph** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac66c>)

Move cursor forward one paragraph (to next all-whitespace line).

**forward-paragraph-extend** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac6ac>)

Move cursor forward one paragraph (to next all-whitespace line), adjusting the selection range to new position.

**forward-tab** ()

Ein Tab-Zeichen an der aktuellen Cursor-Position setzen

**forward-word** (delimiters=None, gravity='start', repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac5ec>)

Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be „start“ or „end“ to indicate whether cursor is placed at start or end of the word.

**forward-word-extend** (delimiters=None, gravity='start', repeat=<command.commandmgr.kArgNumeric instance at 0x412ac60c>)

Move cursor forward one word, adjusting the selection range to new position. Optionally, rovide a string that contains the delimiters to define which characters are part of a word. Gravity may be „start“ or „end“ to indicate whether cursor is placed at start or end of the word.

**hide-selection** ()

Anzeige der aktuellen Textauswahl ausschalten

**indent-to-match** ()

Die aktuelle Zeile oder den gewählten Bereich einrücken, um sie an die Einrückung der vorhergehenden nicht-leeren Zeile anzupassen

**indent-to-next-indent-stop** ()

Indent to next indent stop from the current position. Acts like indent command if selection covers multiple lines.

**isearch-backward** (search_string=None, repeat=<command.commandmgr.kArgNumericModifier instance at 0x412a1fec>)

Eine inkrementale Mini-Suche rückwärts von der aktuellen Position beginnen, die gegebene Suchzeichenkette wahlweise eingeben

**isearch-backward-regex** (search_string=None, repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac04c>)

Initiate incremental regular expression mini-search backward from the cursor position, optionally entering the given search string

**isearch-forward** (search_string=None, repeat=<command.commandmgr.kArgNumericModifier instance at 0x412a166c>)

Eine inkrementale Mini-Suche vorwärts von der aktuellen Position beginnen, die gegebene Suchzeichenkette wahlweise eingeben

**isearch-forward-regex** (search_string=None, repeat=<command.commandmgr.kArgNumericM instance at 0x412a1fcc>)

Initiate incremental regular expression mini-search forward from the cursor position, optionally entering the given search string

**isearch-repeat** (reverse=False, repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac08c>)

Repeat the most recent isearch, using same string and regex/text. Reverse direction when reverse is True.

**isearch-sel-backward** (persist=True, whole_word=False, repeat=<command.commandmgr.kAr instance at 0x412ac14c>)

Initiate incremental mini-search backward from the cursor position, using current selection as the search string. Set persist=False to do the search but end the interactive search session immediately.

**isearch-sel-forward** (persist=True, whole_word=False, repeat=<command.commandmgr.kArgN instance at 0x412ac10c>)

Initiate incremental mini-search forward from the cursor position, using current selection as the search string. Set persist=False to do the search but end the interactive search session immediately.

**kill-line** ()

Rest der Zeile vom Cursor bis zum Zeilenende löschen und es mit allen anderen zusammenhängend entfernten Zeilen in die Zwischenablage platzieren. Das Ende der Zeile wird nur entfernt, wenn nichts zwischen dem Cursor und dem Zeilenende ist.

**middle-of-screen-line** ()

Move to middle of current wrapped line

**middle-of-screen-line-extend** ()

Move to middle of current wrapped line, extending selection

**move-to-register** (unit='char', cut=0, num=<command.commandmgr.kArgNumericModifier instance at 0x412a16ec>)

Cut or copy a specified number of characters or lines, or the current selection. Set cut=1 to remove the range of text from the editor after moving to register (otherwise it is just copied). Unit should be one of 'char' or 'line' or 'sel' for current selection.

**move-to-register-next-move** (cut=0, repeat=<command.commandmgr.kArgNumericModifier instance at 0x412a1f6c>)

Move the text spanned by the next cursor motion to a register

**new-line** ()

Eine neue Zeile an der aktuellen Cursor-Position setzen

**next-line** (cursor='same', repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac2ac>)

Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**next-line-extend** (cursor='same', repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac2cc>)

Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**next-line-extend-rect** (cursor='same', repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac30c>)

Move to next screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**next-line-in-file** (cursor='start', repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac40c>)

Move to next line in file, repositioning character within line: 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**paste** ()

Text aus der Zwischenablage einfügen

**paste-register** (pos=1, indent=0, cursor=-1)

Paste text from register as before or after the current position. If the register contains only lines, then the lines are pasted before or after current line (rather than at cursor). If the register contains fragments of lines, the text is pasted over the current selection

or either before or after the cursor. Set pos = 1 to paste after, or -1 to paste before. Set indent=1 to indent the pasted text to match current line. Set cursor=-1 to place cursor before lines or cursor=1 to place it after lines after paste completes.

**previous-line** (cursor='same', repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac34c>)

Move to previous screen line, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**previous-line-extend** (cursor='same', repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac38c>)

Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**previous-line-extend-rect** (cursor='same', repeat=<command.commandmgr.kArgNumericMod instance at 0x412ac3cc>)

Move to previous screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**previous-line-in-file** (cursor='start', repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac44c>)

Move to previous line in file, repositioning character within line: 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**profile-editor-start** ()

Editor Auslastungs-Protokollroutine anschalten.

**profile-editor-stop** ()

Editor Auslastungs-Protokollroutine abschalten und Statistik zu stdout drucken

**reanalyze-file** ()

Rescan file for code analysis.

**redo** ()

Letzte Aktion wiederherstellen

**repeat-command** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac1ec>)

Repeat the last editor command

**repeat-search-char** (opposite=0, repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac18c>)

Repeat the last search_char operation, optionally in the opposite direction.

**rstrip-each-line** ()

Strip trailing whitespace from each line.

**scroll-text-down** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412aceec>)

Scroll text down a line w/o moving cursor's relative position on screen. Repeat is number of lines or if >0 and <1.0 then percent of screen.

**scroll-text-left** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412acf2c>)

Scroll text left a column w/o moving cursor's relative position on screen. Repeat is number of columns or if >0 and <1.0 then percent of screen.

**scroll-text-page-down** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ace6c>)

Scroll text down a page w/o moving cursor's relative position on screen. Repeat is number of pages or if >0 and <1.0 then percent of screen.

**scroll-text-page-up** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ace2c>)

Scroll text up a page w/o moving cursor's relative position on screen. Repeat is number of pages or if >0 and <1.0 then percent of screen.

**scroll-text-right** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412acf6c>)

Scroll text right a column w/o moving cursor's relative position on screen. Repeat is number of columns or if >0 and <1.0 then percent of screen.

**scroll-text-up** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412aceac>)

Scroll text up a line w/o moving cursor's relative position on screen. Repeat is number of lines or if >0 and <1.0 then percent of screen.

**scroll-to-cursor** ()

Bild zur aktuellen Cursor-Position rollen, wenn nicht bereits sichtbar

**search-char** (dir=1, pos=0, repeat=<command.commandmgr.kArgNumericModifier instance at 0x412ac16c>, single_line=0)

Search for the given character. Searches to right if dir > 0 and to left if dir < 0. Optionally place cursor pos characters to left or right of the target (e.g., use -1 to place one to left). If repeat > 1, the Nth match is found. Set single_line=1 to search only within the current line.

**select-all** ()

Gesamten Text im Editor auswählen

**set-mark-command** (unit='char')

Set start of text marking for selection at current cursor position. Subsequently, all cursor move operations will automatically extend the text selection until stop-mark-command is issued. Unit defines what is selected: can be one of char, line, or block (rectangle).

**set-register** ()

Set the register to use for subsequent cut/copy/paste operations

**show-autocompleter** ()

Auto-Vervollständiger an der aktuellen Cursor-Position aufschlagen

**show-selection** ()

Anzeige der aktuellen Textauswahl anschalten

**start-of-document** ()

Cursor zum Anfang des Dokuments bewegen

**start-of-document-extend** ()

Cursor zum Anfang des Dokuments bewegen und den Auswahllbereich an die neue Position anpassen

**stop-mark-command** (deselect=True)

Stop text marking for selection at current cursor position, leaving the selection set as

is. Subsequent cursor move operations will deselect the range and set selection to cursor position. Deselect immediately when deselect is True.

**undo** ()

Letzte Aktion rückgangig machen

**yank-line** ()

Inhalte des Kill-Buffers, die mit Zeile-löschen im Bearbeiten-Puffer erstellt wurden, ziehen

# General Editor Commands

Editor-Befehle, die für den aktuellen (zuletzt aktiven) Source-Editor gelten, egal ob für diesen die Tastatur aktiviert ist.

**check-indent-consistency** ()

Überprüfen, ob Einrückungen konsequent Leerzeichen oder Tabs durch die gesamte Datei verwenden.

**comment-out-region** ()

Gewählten Bereich auskommentieren

**convert-indents-to-mixed** ()

Alle Zeilen mit führenden Leerzeichen in gemischte Tabs und Leerzeichen umwandeln.

**convert-indents-to-spaces-only** ()

Alle Zeilen, die führende Tabs enthalten, in Nur-Leerzeichen umwandeln.

**convert-indents-to-tabs-only** ()

Alle Einrückungen in die Verwendung von nur Tab-Zeichen und keine Leerzeichen umwandeln.

**execute-kbd-macro** (register='a', repeat=<command.commandmgr.kArgNumericModifier instance at 0x412b222c>)

Execute most recently recorded keyboard macro. If register is None then the user is asked to enter a letter a-z for the register where the macro is filed. Otherwise, register 'a' is used by default.

**fill-paragraph** ()

Versuch, den Paragraphen um den aktuellen Beginn der Auswahl automatisch auszurichten

**fold-collapse-all** ()

Alle Faltepunkte in der aktuellen Datei zusammenklappen

**fold-collapse-all-clicked** ()

Collapse the clicked fold point completely

**fold-collapse-all-current** ()

Den aktuellen Faltepunkt vollständig zusammenklappen

**fold-collapse-more-clicked** ()

Collapse the clicked fold point one more level

**fold-collapse-more-current** ()

Den aktuellen Faltepunkt um ein weiteres Level zusammenklappen

**fold-expand-all** ()

Alle Faltepunkte in der aktuellen Datei erweitern

**fold-expand-all-clicked** ()

Expand the clicked fold point completely

**fold-expand-all-current** ()

Den aktuellen Faltepunkt vollständig erweitern

**fold-expand-more-clicked** ()

Expand the clicked fold point one more level

**fold-expand-more-current** ()

Den aktuellen Faltepunkt um ein weiteres Level erweitern

**fold-toggle** ()

Den aktuellen Faltepunkt wechseln

**fold-toggle-clicked** ()

Toggle the clicked fold point

**force-indent-style-to-match-file** ()

Den Einrückungsstil des Editors zwingen, den Einrückungsstil an denjenigen anzupassen, der in der Mehrheit der Datei gefunden wurde

**force-indent-style-to-mixed** ()

Den Einrückungsstil des Editors zum gemischten Gebrauch von Tabs und Leerzeichen zwingen, ohne Beachtung der Inhalte der Datei

**force-indent-style-to-spaces-only** ()

Den Einrückungsstil des Editors zur Verwendung von Nur-Leerzeichen zwingen, ohne Beachtung der Inhalte der Datei

**force-indent-style-to-tabs-only** ()

Den Einrückungsstil des Editors zur Verwendung von Nur-Tabs zwingen, ohne Beachtung der Inhalte der Datei

**goto-clicked-symbol-defn** ()

Zur Definition des Source-Symbols gehen, auf das zuletzt geklickt wurde

**goto-column** (column=<command.commandmgr.kArgNumericModifier instance at 0x412b21cc>)

Move cursor to given column

**goto-line** (lineno=<command.commandmgr.kArgNumericModifier instance at 0x412b20cc>)

Cursor am Anfang der gegebenen Zeilennummer positionieren

**goto-nth-line** (lineno=<command.commandmgr.kArgNumericModifier instance at 0x412b210c>, cursor='start')

Position cursor at start of given line number (1=first, -1 = last). This differs from goto-line in that it never prompts for a line number but instead uses the previously entered numeric modifier or defaults to going to line one. The cursor can be positioned at 'start', 'end', or 'fnb' for first non-blank character.

**goto-nth-line-default-end** (lineno=<command.commandmgr.kArgNumericModifier instance at 0x412b214c>, cursor='start')

Same as goto_nth_line but defaults to end of file if no lineno is given

**goto-percent-line** (percent=<command.commandmgr.kArgNumericModifier instance at 0x412b218c>, cursor='start')

Position cursor at start of line at given percent in file. This uses the previously entered numeric modifier or defaults to going to line one. The cursor can be positioned at 'start', 'end', or 'fnb' for first non-blank character, or in VI mode it will do brace matching operation to reflect how VI overrides this command.

**goto-selected-symbol-defn** ()

Zur Definition des gewählten Source-Symbols gehen

**hide-all-whitespace** ()

Alle speziellen Markierungen ausschalten, um Leerräume und das Zeilenende anzuzeigen

**hide-eol** ()

Spezielle Markierungen ausschalten, um die Zeichen am Zeilenende anzuzeigen

**hide-indent-guides** ()

Spezielle Markierungen ausschalten, um das Einrückungslevel anzuzeigen

**hide-whitespace** ()

Spezielle Markierungen ausschalten, um Leeräume anzuzeigen

**indent-lines** (num=<command.commandmgr.kArgNumericModifier instance at 0x412b22ac>)

Indent selected number of lines from cursor position

**indent-next-move** (num=<command.commandmgr.kArgNumericModifier instance at 0x412b234c>)

Indent lines spanned by next cursor move

**indent-region** (sel=None)

Indent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or „never-select“ to unselect after indent, „always-select“ to always select after indent, or „retain-select“ to retain current selection after indent.

**indent-to-match-next-move** (num=<command.commandmgr.kArgNumericModifier instance at 0x412b23cc>)

Indent lines spanned by next cursor move to match, based on the preceding line

**insert-command** (cmd)

Insert the output for the given command at current cursor position. Some special characters in the command line (if not escaped with ) will be replaced as follows:

Unexpected indentation.

> % -- Current file's full path name # -- Previous file's full path name

**insert-file** (filename)

Eine Datei an der aktuellen Cursor-Position einfügen, fordert Nutzer zur Dateiauswahl auf

**join-lines** (delim=' ', num=<command.commandmgr.kArgNumericModifier instance at 0x412b242c>)

Join together specified number of lines after current line (replace newlines with the given delimiter (single space by default)

**join-selection** (delim=' ')

Join together all lines in given selection (replace newlines with the given delimiter (single space by default)

**kill-buffer** ()

Aktuelle Textdatei schließen

**outdent-lines** (num=<command.commandmgr.kArgNumericModifier instance at 0x412b230c>)

Outdent selected number of lines from cursor position

**outdent-next-move** (num=<command.commandmgr.kArgNumericModifier instance at 0x412b238c>)

Outdent lines spanned by next cursor move

**outdent-region** (sel=None)

Outdent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or „never-select“ to unselect after indent, „always-select“ to always select after indent, or „retain-select“ to retain current selection after indent.

**page-setup** ()

Einstellungsdialog der Druckseite anzeigen

**print-view** ()

Aktives Editor-Dokument drucken

**query-replace** (search_string, replace_string)

Eine inkrementale Mini-Suche Anfrage/Ersetzen von der Cursor- Position beginnen.

**query-replace-regex** (search_string, replace_string)

Initiate incremental mini-search query/replace from the cursor position. The search string is treated as a regular expression.

**range-replace** (search_string, replace_string, confirm, range_limit, match_limit, regex)

Initiate incremental mini-search query/replace within the given selection. This is similar to query_replace but allows some additional options: confirm -- True to confirm each replace range_limit -- None to replace between current selection start and end of document,

Unexpected indentation.

> 1 to limit operation to current selection or to current line of selection is empty, (start, end) to limit operation to within given selection range, or „first|last“ to limit operating withing given range of lines.

Block quote ends without a blank line; unexpected unindent.

match_limit -- None to replace any number of matches, or limit of number of replaces regex -- Treat search string as a regular expression

**repeat-replace** (repeat=<command.commandmgr.kArgNumericModifier instance at 0x412b206c>)

Repeat the last query replace or range replace operation on the current line. The first match is replaced without confirmation.

**replace-char** (line_mode='multiline', num=<command.commandmgr.kArgNumericModifier instance at 0x412b20ac>)

Replace num characters with given character. Set line_mode to multiline to allow replacing across lines, extend to replace on current line and then extend the line length, and restrict to replace only if enough characters exist on current line after cursor position.

**replace-string** (search_string, replace_string)

Alle Vorkommen einer Zeichenkette von der Cursor-Position bis zum Ende der Datei ersetzen.

**replace-string-regex** (search_string, replace_string)

Replace all occurrences of a string from the cursor position to end of file. The search string is treated as a regular expression.

**save-buffer** ()

Aktuelle Textdatei auf dem Laufwerk speichern

**set-readonly** ()

Editor auf nur-lesbar setzen. Diese kann nicht gemacht werden wann der Editor ungespeicherte Änderungen enthält.

**set-writable** ()

Editor auf schreibbar setzen. Diese kann benutzt werden um den nur-lesbaren Zustand des Editors zu ändern, wenn ein nur-lesbares Datei geöffnet wird.

**show-all-whitespace** ()

Alle speziellen Markierungen anschalten, um Leerräume und das Zeilenende anzuzeigen

**show-eol** ()

Spezielle Markierungen anschalten, um die Zeichen am Zeilenende anzuzeigen

**show-indent-guides** ()

Spezielle Markierungen anschalten, um das Einrückungslevel anzuzeigen

**show-indent-manager** ()

Den Einrückungsmanager für diese Editordatei anzeigen

**show-whitespace** ()

Spezielle Markierungen anschalten, um Leerräume anzuzeigen

**start-kbd-macro** (register='a')

Start definition of a keyboard macro. If register=None then the user is prompted to enter a letter a-z under which to file the macro. Otherwise, register 'a' is used by default.

**stop-kbd-macro** ()

Definition eines Tastatur-Makros stoppen

**toggle-line-wrapping** ()

Toggles line wrapping preference for all editors

**toggle-overtype** ()

Status Überschreibmodus wechseln

**uncomment-out-region** ()

Kommentar im gewählten Bereich aufheben

**use-lexer-ada** ()

Syntax-Markierung für Ada-Source erzwingen

**use-lexer-apache-conf** ()

Syntax-Markierung für Apache-Konfigurationsdateiformat erzwingen

**use-lexer-asm** ()

Syntax-Markierung für die Masm Assemblersprache erzwingen

**use-lexer-ave** ()

Syntax-Markierung für Avenue GIS-Sprache erzwingen

**use-lexer-baan** ()

Syntax-Markierung für Baan erzwingen

**use-lexer-bash** ()

Syntax-Markierung für Bash-Skripte erzwingen

**use-lexer-bullant** ()

Syntax-Markierung für Bullant erzwingen

**use-lexer-by-doctype** ()

Syntax-Markierung entsprechend dem Dateityp verwenden

**use-lexer-cpp** ()

Syntax-Markierung für C/C++-Source erzwingen

**use-lexer-css2** ()

Syntax-Markierung für CSS2 erzwingen

**use-lexer-diff** ()

Syntax-Markierung für diff/cdiff-Dateien erzwingen

**use-lexer-dos-batch** ()

Syntax-Markierung für DOS Batch-Dateien erzwingen

**use-lexer-eiffel** ()

Syntax-Markierung für Eiffel-Source erzwingen

**use-lexer-errlist** ()

Syntax-Markierung für Fehlerlisten-Format erzwingen

**use-lexer-escript** ()

Syntax-Markierung für EScript erzwingen

**use-lexer-fortran** ()

Syntax-Markierung für Fortran erzwingen

**use-lexer-html** ()

Syntax-Markierung für HTML erzwingen

**use-lexer-idl** ()

Syntax-Markierung für XP IDL erzwingen

**use-lexer-java** ()

Syntax-Markierung für Java-Source erzwingen

**use-lexer-javascript** ()

Syntax-Markierung für Javascript erzwingen

**use-lexer-latex** ()

Syntax-Markierung für LaTeX erzwingen

**use-lexer-lisp** ()

Syntax-Markierung für Lisp-Source erzwingen

**use-lexer-lout** ()

Syntax-Markierung für die LOUT Typesetting-Sprache erzwingen

**use-lexer-lua** ()

Syntax-Markierung für Lua erzwingen

**use-lexer-makefile** ()

Syntax-Markierung für Makefiles erzwingen

**use-lexer-matlab** ()

Syntax-Markierung für Matlab erzwingen

**use-lexer-mmixal** ()

Syntax-Markierung für die MMIX Assemblersprache erzwingen

**use-lexer-msidl** ()

Syntax-Markierung für MS IDL erzwingen

**use-lexer-nncrontab** ()

Syntax-Markierung für NNCrontab-Dateien erzwingen

**use-lexer-none** ()

Keine Syntax-Markierung verwenden

**use-lexer-nsis** ()

Syntax-Markierung für NSIS erzwingen

**use-lexer-pascal** ()

Syntax-Markierung für Pascal-Source erzwingen

**use-lexer-perl** ()

Syntax-Markierung für Perl-Source erzwingen

**use-lexer-php** ()

Syntax-Markierung für PHP-Source erzwingen

**use-lexer-plsql** ()

Syntax-Markierung für PL/SQL-Dateien erzwingen

**use-lexer-pov** ()

Syntax-Markierung für die POV Ray Tracer Scene Beschreibungssprache erzwingen

**use-lexer-properties** ()

Syntax-Markierung für Eigenschaftsdateien erzwingen

**use-lexer-ps** ()

Syntax-Markierung für Postscript erzwingen

**use-lexer-python** ()

Syntax-Markierung für Python-Source erzwingen

**use-lexer-rc** ()

Syntax-Markierung für RC-Dateiformat erzwingen

**use-lexer-ruby** ()

Syntax-Markierung für Ruby-Source erzwingen

**use-lexer-scriptol** ()

Syntax-Markierung für Scriptol erzwingen

**use-lexer-sql** ()

Syntax-Markierung für SQL erzwingen

**use-lexer-tcl** ()

Syntax-Markierung für TCL erzwingen

**use-lexer-vb** ()

Syntax-Markierung für Visual Basic-Source erzwingen

**use-lexer-vxml** ()

Syntax-Markierung für VXML erzwingen

**use-lexer-xcode** ()

Syntax-Markierung für XCode-Dateien erzwingen

**use-lexer-xml** ()

Syntax-Markierung für XML-Dateien erzwingen

**use-lexer-yaml** ()

Syntax-Markierung für YAML erzwingen

**zoom-in** ()

Vergrößern, vergrößert die Größe der Textanzeige vorübergehend um einen Schriftgrad

**zoom-out** ()

Verkleinern, verkleinert die Größe der Textanzeige vorübergehend um eine Schriftgröße

# Project Manager Commands

Diese Befehle sind für den Projektmanager oder das aktuelle Projekt, ungeachtet der Tatsache, ob die Tastatur für die Projektliste aktiviert ist.

**add-current-file-to-project** ()

Die vorderste der gegenwärtig geöffneten Dateien zum Projekt hinzufügen

**add-file-to-project** ()

Eine bestehende Datei zum Projekt hinzufügen.

**add-package-to-project** ()

Ein Paket zum Projekt hinzufügen.

**add-tree-to-project** ()

Einen ganzen Verzeichnisbaum zum Projekt hinzufügen.

**browse-selected-from-project** ()

Die gegenwärtig ausgewählte Datei im Projektmanager durchsuchen

**clear-project-main-debug-file** ()

Die Haupt-Debug-Datei löschen, so dass das Debuggen standardmäßig im vordersten Fenster ausgeführt wird.

**close-project** ()

Gegenwärtig geöffnete Projektdatei schließen

**compact-project** ()

Verdichten der gegenwärtig geöffneten Projektdatei durch das Entfernen von Informationen über nicht bestehende Dateien und nicht-kritische Attribute für Dinge wie visueller Status.

**debug-selected-from-project** ()

Debuggen der gegenwärtig ausgewählten Datei im Projektmanager starten

**execute-selected-from-project** ()

Die gegenwärtig ausgewählte Datei im Projektmanager ausführen

**new-project** ()

Ein neues Projekt anlegen.

**open-ext-selected-from-project** ()

Die gegenwärtig ausgewählte Datei im Projektmanager öffnen

**open-project** ()

Eine Projektdatei öffnen.

**open-selected-from-project** ()

Die gegenwärtig im Projektmanager ausgewählten Dateien öffnen

**remove-selection-from-project** ()

Die gegenwärtig ausgewählte Datei oder das Paket vom Projekt entfernen.

**save-project** ()

Projektdatei speichern.

**save-project-as** ()

Projektdatei unter einem anderen Namen speichern.

**set-current-as-main-debug-file** ()

Die gegenwärtig vorderste Datei als Haupt-Debug-Datei für dieses Projekt einstellen

**set-selected-as-main-debug-file** ()

Die gewählte Datei als Haupt-Debug-Datei für dieses Projekt einstellen

**show-analysis-stats** ()

Statistiken der Source-Code-Analyse anzeigen

**show-project-window** ()

Das Fenster des Projektmanagers aufschlagen

**use-normal-project** ()

Projekt in normalem Format speichern

**use-shared-project** ()

Projekt in gemeinsam nutzbaren Format speichern

**view-file-properties** (loc=None)

Projekteigenschaften für eine bestimmte Datei (aktuelle Datei wenn nicht gegeben)

**view-project-as-flat-tree** ()

Projekt als abgeflachten Verzeichnisbaum von der Projektdatei anzeigen

**view-project-as-tree** ()

Projekt als Verzeichnisbaum von der Projektdatei anzeigen

**view-project-by-mime-type** ()

Projekt als Baum, der nach Datei-Mime-Typ organisiert ist, anzeigen

**view-project-properties** (highlighted_attrib=None)

Projektweite Eigenschaften ansehen oder ändern

# Project View Commands

Befehle, die nur zur Verfügung stehen, wenn die Tastatur für die Projektansicht aktiviert ist.

**browse-selected-from-project** ()

Die gegenwärtig ausgewählte Datei im Projektmanager durchsuchen

**debug-selected-from-project** ()

Debuggen der gegenwärtig ausgewählten Datei im Projektmanager starten

**execute-selected-from-project** ()

Die gegenwärtig ausgewählte Datei im Projektmanager ausführen

**open-ext-selected-from-project** ()

Die gegenwärtig ausgewählte Datei im Projektmanager öffnen

**open-selected-from-project** ()

Die gegenwärtig im Projektmanager ausgewählten Dateien öffnen

**remove-selection-from-project** ()

Die gegenwärtig ausgewählte Datei oder das Paket vom Projekt entfernen.

**set-selected-as-main-debug-file** ()

Die gewählte Datei als Haupt-Debug-Datei für dieses Projekt einstellen

**view-project-as-flat-tree** ()

Projekt als abgeflachten Verzeichnisbaum von der Projektdatei anzeigen

**view-project-as-tree** ()

Projekt als Verzeichnisbaum von der Projektdatei anzeigen

**view-project-by-mime-type** ()

Projekt als Baum, der nach Datei-Mime-Typ organisiert ist, anzeigen

# Debugger Commands

Befehle für den Debugger und den aktuellen Debug-Prozess, wenn vorhanden.

**break-clear** ()

Einen Haltepunkt an der aktuellen Zeile löschen

**break-clear-all** ()

Alle Haltepunkte löschen

**break-clear-clicked** ()

Einen Haltepunkt am Ort des Mausklicks löschen

**break-disable** ()

Einen Haltepunkt an der aktuellen Zeile deaktivieren

**break-disable-all** ()

Disable all breakpoints

**break-disable-clicked** ()

Einen Haltepunkt am Ort des Mausklicks deaktivieren

**break-edit-cond** ()

Bedingung für Haltepunkt auf der aktuellen Zeile bearbeiten

**break-edit-cond-clicked** ()

Edit condition for the breakpoint at the current mouse click location

**break-enable** ()

Einen Haltepunkt an der aktuellen Zeile aktivieren

**break-enable-all** ()

Enable all breakpoints

**break-enable-clicked** ()

Einen Haltepunkt am Ort des Mausklicks aktivieren

**break-enable-toggle** ()

Schaltet ein, ob der Haltepunkt an der aktuellen Zeile aktiviert oder deaktiviert ist

**break-ignore** ()

Den Haltepunkt auf der aktuellen Zeile für N Iterationen ignorieren

**break-ignore-clicked** ()

Ignore the breakpoint at the current mouse click location for N iterations

**break-set** ()

Einen neuen, regulären Haltepunkt auf der aktuelle Zeile setzen

**break-set-clicked** ()

Einen neuen, regulären Haltepunkt am Ort des Mausklicks setzen

**break-set-cond** ()

Einen neuen, bedingten Haltepunkt setzen an der aktuellen Zeile

**break-set-cond-clicked** ()

Set a new conditionalbreakpoint at the current mouse click location

**break-set-temp** ()

Einen neuen, temporären Haltepunkt setzen auf der aktuellen Zeile

**break-set-temp-clicked** ()

Set a new temporary breakpoint at the current mouse click location

**break-toggle** ()

Haltepunkt an der aktuellen Zeile umschalten (erstellt neuen regulären Haltepunkt, wenn einer erstellt ist)

**clear-exception-ignores-list** ()

Liste der Exceptions, die während des Debuggens ignoriert wurden, löschen

**clear-var-errors** ()

Gespeicherte Variablenfehler löschen, so dass sie erneut abgerufen werden

**collapse-tree-more** ()

Ansicht aller gewählten Variablen eine weitere Ebene zusammenklappen

**debug-attach** ()

Zu einem bereits laufenden Debug-Prozess hinzufügen

**debug-continue** ()

Ausführung fortfahren (oder starten), zum nächsten Haltepunkt

**debug-detach** ()

Vom Debug-Prozess abtrennen und ausführen lassen

**debug-file** ()

Debuggen der aktuellen Datei starten (anstelle des Debug-Startpunktes)

**debug-kill** ()

Debuggen stoppen

**debug-stop** ()

Frei-laufende Ausführung an aktuellem Programmzähler anhalten

**exception-always-stop** ()

Immer an Exceptions anhalten, selbst wenn sie im Code abgefangen werden

**exception-never-stop** ()

Niemals an Exceptions anhalten, selbst wenn sie nicht im Code abgefangen werden

**exception-unhandled-stop** ()

Nur an Exceptions stoppen, die nicht im Code abgefangen werden

**expand-tree-more** ()

Ansicht aller gewählten Variblen einen Ast tiefer erweitern

**force-var-reload** ()

Erneutes Abrufen eines Wertes vom Server erzwingen

**frame-down** ()

Aktuellen Debug-Stack nach unten gehen

**frame-up** ()

Aktuellen Debug-Stack nach oben gehen

**hide-detail** ()

Show the textual value detail area

**run-to-cursor** ()

Zur aktuellen Cursor-Position gehen

**show-detail** ()

Show the textual value detail area

**step-into** ()

In den aktuellen Ausführungspunkt gehen oder Debuggen an der ersten Zeile starten

**step-out** ()

Von aktueller Funktion zurückkehren

**step-over** ()

Über den aktuellen Ausführungspunkt schreiten

**watch** (style='ref')

Gewählte Variable beobachten unter Verwendung eines direkten Objektverweises, um sie zu verfolgen

**watch-expression** (expr=None)

Einen neuen Ausdruck zur Beobachtungsliste hinzufügen

**watch-module-ref** ()

Markierten Wert relativ zu einem Modul, das nach Namen in sys.modules nachgeschlagen wurde beobachten

**watch-parent-ref** ()

Gewählte Variable beobachten unter Verwendung eines Verweises zum Parent des Wertes und dem Key-Slot des Wertes

**watch-ref** ()

Gewählte Variable beobachten unter Verwendung eines direkten Objektverweises, um sie zu verfolgen

**watch-symbolic** ()

Gewählten Wert beobachten unter Verwendung des symbolischen Pfades zu ihm

# Debugger Watch Commands

Befehle für das Beobachten-Werkzeug des Debuggers (nur für Wing IDE Professional). Diese sind nur verfügbar, wenn die Tastatur für das Beobachten-Werkzeug aktiviert ist.

**watch-clear-all** ()

Alle Einträge von der Beobachtungsliste löschen

**watch-clear-selected** ()

Markierten Eintrag von der Beobachtungsliste löschen

# Lizenzinformationen

Wing IDE ist ein kommerzielles Produkt, das auf einer Reihe von Open Source Technologien basiert. Obwohl der Source-Code des Produktes für Nutzer von Wing IDE Professional zur Verfügung steht (mit Unterzeichnung einer Geheimhaltungsvereinbarung), ist das Produkt selbst nicht Open Source.

Die folgenden Abschnitte beschreiben die Lizenzierung für das Produkt als Ganzes (Endnutzervereinbarung) und stellen die erforderlichen Legal Statements für die enthaltenen Open Source Komponenten bereit.

## 10.1. Wing IDE Software-Lizenz

This End User License Agreement (EULA) is a CONTRACT between you (either an individual or a single entity) and Wingware, which covers your use of either „Wing IDE Professional" or „Wing IDE Enterprise" and related software components. All such software is referred to herein as the „Software Product." A software license and a license key or serial number („Software Product License"), issued to a designated user only by Wingware or its authorized agents, is required for each concurrent user of the Software Product. If you do not agree to the terms of this EULA, then do not install or use the Software Product or the Software Product License. By explicitly accepting this EULA you are acknowledging and agreeing to be bound by the following terms:

1. EVALUATION LICENSE WARNING

This Software Product can be used in conjunction with a free evaluation Software Product License. If you are using such an evaluation Software Product License, you may use the Software Product only to evaluate its suitability for purchase. Evaluation Software Product Licenses have an expiration date and most of the features of the software will be disabled after that date. WINGWARE BEARS NO LIABILITY FOR ANY DAMAGES RESULTING FROM USE (OR ATTEMPTED USE AFTER THE EXPIRATION DATE) OF THE SOFTWARE PRODUCT, AND HAS NO DUTY TO PROVIDE ANY

SUPPORT BEFORE OR AFTER THE EXPIRATION DATE OF AN EVALUATION LICENSE.

2. GRANT OF NON-EXCLUSIVE LICENSE

Wingware grants the non-exclusive, non-transferable right for a single user to use this Software Product on a single operating system per license purchased. Each additional concurrent user of the Software Product, or each additional operating system where the product is used, requires an additional Software Product License. This includes operating systems on which the Software Product is compiled from source code by the user.

Wingware grants you the right to modify, alter, improve, or enhance the Software Product without limitation, except as described in this EULA.

Although rights to modification of the Software Product are granted by this EULA, you may not tamper with, alter, or use the Software Product in a way that disables, circumvents, or otherwise defeats its built-in licensing verification and enforcement capabilities. The right to modification of the Software Product also does not include the right to remove or alter any trademark, logo, copyright or other proprietary notice, legend, symbol or label in the Software Product.

You may at your discretion distribute patch files containing any modifications or improvements made to the Software Product, other than those that are aimed at disabling or circumventing its built-in license verification capabilities, or that result in the removal or alteration of any trademark, logo, copyright, or other proprietary notice, legend, symbol or label in the Software Product. This right does not include the right to distribute substantial portions of the original source, where distribution rights are limited to contextual information normally existing in software patch files.

You may at your discretion designate license terms, open source or otherwise, for all modifications or improvements made by you. Wingware has no special rights to any such modifications or improvements.

You may make copies of the Software Product as reasonably necessary for its use. Each copy must reproduce all copyright and other proprietary rights notices on or in the Software Product.

You may install each Software Product License on a single computer system. A second installation of the same Software Product License may be made on one other computer system, so long as both copies of the same Software Product License never come into concurrent use. You may also make copies of the Software Product License as necessary for backup and/or archival purposes. Backup and archival copies may not come into active use, together with the Software Product, for any purpose. No other copies may be made. Each copy must reproduce all copyright and other proprietary rights notices

on or in the Software Product License. You may not modify or create derivative copies of the Software Product License.

All rights not expressly granted to you are retained by WIngware.

3. INTELLECTUAL PROPERTY RIGHTS RESERVED BY WINGWARE

The Software Product is owned by Wingware and is protected by United States and international copyright laws and treaties, as well as other intellectual property laws and treaties. You must not remove or alter any copyright notices on any copies of the Software Product. This Software Product copy is licensed, not sold. You may not use, copy, or distribute the Software Product, except as granted by this EULA, without written authorization from Wingware or its designated agents. Furthermore, this EULA does not grant you any rights in connection with any trademarks or service marks of Wingware. Wingware reserves all intellectual property rights, including copyrights, and trademark rights.

4. NO RIGHT TO TRANSFER

You may not rent, lease, lend, or in any way distribute or transfer any rights in this EULA or the Software Product to third parties without Wingware's written approval, and subject to written agreement by the recipient of the terms of this EULA.

5. INDEMNIFICATION

You hereby agree to indemnify Wingware against and hold harmless Wingware from any claims, lawsuits or other losses that arise out of your breach of any provision of this EULA.

6. THIRD PARTY RIGHTS

Any software provided along with the Software Product that is associated with a separate license agreement is licensed to you under the terms of that license agreement. This license does not apply to those portions of the Software Product. Copies of these third party licenses are included in all copies of the Software Product.

7. SUPPORT SERVICES

Wingware may provide you with support services related to the Software Product. Use of any such support services is governed by Wingware policies and programs described in online documentation and/or other Wingware-provided materials.

As part of these support services, Wingware may make available bug lists, planned feature lists, and other supplemental informational materials. WINGWARE MAKES NO WARRANTY OF ANY KIND FOR THESE MATERIALS AND ASSUMES NO LIABILITY WHATSOEVER FOR DAMAGES RESULTING FROM ANY USE OF THESE MATERIALS. FURTHERMORE, YOU MAY NOT USE ANY MATERIALS PROVIDED IN THIS WAY TO SUPPORT ANY CLAIM MADE AGAINST WINGWARE.

Any supplemental software code or related materials that Wingware provides to you as part of the support services, in periodic updates to the Software Product or otherwise, is to be considered part of the Software Product and is subject to the terms and conditions of this EULA.

With respect to any technical information you provide to Wingware as part of the support services, Wingware may use such information for its business purposes without restriction, including for product support and development. Wingware will not use such technical information in a form that personally identifies you without first obtaining your permission.

9. TERMINATION WITHOUT PREJUDICE TO ANY OTHER RIGHTS

Wingware may terminate this EULA if you fail to comply with any term or condition of this EULA. In such event, you must destroy all copies of the Software Product and Software Product Licenses.

10. U.S. GOVERNMENT USE

If the Software Product is licensed under a U.S. Government contract, you acknowledge that the software and related documentation are „commercial items,“ as defined in 48 C.F.R 2.01, consisting of „commercial computer software“ and „commercial computer software documentation,“ as such terms are used in 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1. You also acknowledge that the software is „commercial computer software“ as defined in 48 C.F.R. 252.227-7014(a)(1). U.S. Government agencies and entities and others acquiring under a U.S. Government contract shall have only those rights, and shall be subject to all restrictions, set forth in this EULA. Contractor/manufacturer is Wingware, P.O. Box 1937, Brookline MA 02446-0016, USA.

11. EXPORT RESTRICTIONS

You will not download, export, or re-export the Software Product, any part thereof, or any software, tool, process, or service that is the direct product of the Software Product, to any country, person, or entity -- even to foreign units of your own company -- if such a transfer is in violation of U.S. export restrictions.

12. NO WARRANTIES

YOU ACCEPT THE SOFTWARE PRODUCT AND SOFTWARE PRODUCT LICENSE „AS IS," AND WINGWARE AND ITS THIRD PARTY SUPPLIERS AND LICENSORS MAKE NO WARRANTY AS TO ITS USE, PERFORMANCE, OR OTHERWISE. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, WINGWARE AND ITS THIRD PARTY SUPPLIERS AND LICENSORS DISCLAIM ALL OTHER REPRESENTATIONS, WARRANTIES, AND CONDITIONS, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NON-INFRINGEMENT. THE ENTIRE RISK ARISING OUT OF USE OR PERFORMANCE OF THE SOFTWARE PRODUCT REMAINS WITH YOU.

13. LIMITATION OF LIABILITY

THIS LIMITATION OF LIABILITY IS TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW. IN NO EVENT SHALL WINGWARE OR ITS THIRD PARTY SUPPLIERS AND LICENSORS BE LIABLE FOR ANY COSTS OF SUBSTITUTE PRODUCTS OR SERVICES, OR FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, OR LOSS OF BUSINESS INFORMATION) ARISING OUT OF THIS EULA OR THE USE OF OR INABILITY TO USE THE SOFTWARE PRODUCT OR THE FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF WINGWARE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, WINGWARE'S, AND ITS THIRD PARTY SUPPLIERS' AND LICENSORS', ENTIRE LIABILITY ARISING OUT OF THIS EULA SHALL BE LIMITED TO THE LESSER OF THE AMOUNT ACTUALLY PAID BY YOU FOR THE SOFTWARE PRODUCT OR THE PRODUCT LIST PRICE; PROVIDED, HOWEVER, THAT IF YOU HAVE ENTERED INTO AN WINGWARE SUPPORT SERVICES AGREEMENT, WINGWARE'S ENTIRE LIABILITY REGARDING SUPPORT SERVICES SHALL BE GOVERNED BY THE TERMS OF THAT AGREEMENT.

14. HIGH RISK ACTIVITIES

The Software Product is not fault-tolerant and is not designed, manufactured or intended for use or resale as on-line control equipment in hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of the Software Product, or any software, tool, process, or service that was developed using the Software Product, could lead directly to death,

personal injury, or severe physical or environmental damage („High Risk Activities“). Accordingly, Wingware and its suppliers and licensors specifically disclaim any express or implied warranty of fitness for High Risk Activities. You agree that Wingware and its suppliers and licensors will not be liable for any claims or damages arising from the use of the Software Product, or any software, tool, process, or service that was developed using the Software Product, in such applications.

15. GOVERNING LAW; ENTIRE AGREEMENT ; DISPUTE RESOLUTION

This EULA is governed by the laws of the Commonwealth of Massachusetts, U.S.A., excluding the application of any conflict of law rules. The United Nations Convention on Contracts for the International Sale of Goods shall not apply.

This EULA is the entire agreement between Wingware and you, and supersedes any other communications or advertising with respect to the Software Product; this EULA may be modified only by written agreement signed by authorized representatives of you and Wingware.

Unless otherwise agreed in writing, all disputes relating to this EULA (excepting any dispute relating to intellectual property rights) shall be subject to final and binding arbitration in the State of Massachusetts, in accordance with the Licensing Agreement Arbitration Rules of the American Arbitration Association, with the losing party paying all costs of arbitration. Arbitration must be by a member of the American Arbitration Association. If any dispute arises under this EULA, the prevailing party shall be reimbursed by the other party for any and all legal fees and costs associated therewith.

16. GENERAL

If any provision of this EULA is held invalid, the remainder of this EULA shall continue in full force and effect.

A waiver by either party of any term or condition of this EULA or any breach thereof, in any one instance, shall not waive such term or condition or any subsequent breach thereof.

17. OUTSIDE THE U.S.

If you are located outside the U.S., then the provisions of this Section shall apply. Les parties aux présentes confirment leur volonté que cette convention de même que tous les documents y compris tout avis qui s'y rattache, soient redigés en langue anglaise. (translation: „The parties confirm that this EULA and all related documentation is and will be in the English language.“) You are responsible for complying with any local laws in your jurisdiction which might impact your right to import, export or use the Software

Product, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

18. TRADEMARKS

The following are trademarks or registered trademarks of Wingware: Wingware, the dancing bird logo, Wing IDE, Wing IDE Personal, Wing IDE Professional, Wing IDE Enterprise, Wing Debugger, and „Take Flight!“.

19. CONTACT INFORMATION

If you have any questions about this EULA, or if you want to contact Wingware for any reason, please direct all correspondence to: Wingware, P.O. Box 1937, Brookline, MA 02446-0016, United States of America or send email to info at wingware.com.

## 10.2. Open Source Lizenzinformationen

Wing IDE schließt die folgenden Open Source Technologien ein, von denen die meisten OSI Certified Open Source Lizenzen unterliegen, außer wenn es in den Fußnoten anders ausgewiesen ist:

- atk -- Toolkit für GUI-Zugänglichkeit von Bill.Haneman, Marc.Mulcahy und Padraig.Obriain -- LGPL [1]
- docutils -- reStructuredText Markup-Verarbeitung von David Goodger und Mitarbeitern -- Öffentliche Domäne [2]
- expat -- XML verarbeitende Bibliothek von dem Thai Open Source Software Center Ltd, Clark Cooper und Mitarbeitern -- MIT Lizenz
- fontconfig -- Erkennungsmechanismus und Support für Schriftartkonfiguration von Keith Packard -- MIT Lizenz
- freetype -- Bibliothek für hoch-qualitatives Text-Rendering von Werner Lemberg, David Turner und Mitarbeitern -- FreeType Lizenz
- glib -- Bibliothek für Objektentwicklungssupport von Hans Breuer, Matthias Clasen, Tor Lillqvist, Tim Janik, Havoc Pennington, Ron Steinke, Owen Taylor, Sebastian Wilhelmi und Mitarbeitern -- LGPL [1]

- gtk+ -- GUI-Bibliothek für mehrere Betriebssysteme von Jonathan Blandford, Hans Breuer, Matthias Clasen, Tim Janik, Tor Lillqvist, Federico Mena Quintero, Kristian Rietveld, Søren Sandmann, Manish Singh, Owen Taylor und Mitarbeitern -- LGPL [1]
- gtk-engines -- GTK Theme Engines von The Rasterman, Owen Taylor, Randy Gordon -- LGPL [1]
- gtkscintilla2 -- GTK-Wrapper für Scintilla von Dennis J Houy, Sven Herzberg und Mitarbeitern -- LGPL [1]
- GTK Themen -- Aero von Marcus Petzoldt, LGPL [1]; Aluminum Alloy von Robert Iszaki (roberTO), AluminumAlloy License [4]; Glider von Link Dupont, LGPL [1]; Glossy P von m5brane, nicht spezifiziert [5]; gnububble von Kyle Davis, nicht spezifiziert [5]; H2O von Eric R. Reitz, nicht spezifiziert [5]; High Contrast, Low Contrast, und Large Print themes von Bill Haneman und T. Liebeck, LGPL [1]; Redmond and Redmond95 von Anonymous, nicht spezifiziert [5]; Smokey-Blue von Jakub 'jimmac' Steiner und Paul Hendrick, LGPL [1]; Smooth2000 von ajgenius, nicht spezifiziert [5]; SmoothDesert von Ken Joseph, andere [6]; SmoothRetro von Ken Joseph, andere [6]; SmoothSeaIce von ajgenius, nicht spezifiziert [5]
- gtk-wimp -- GTK-Thema mit nativem Windows Look von Raymond Penners, Evan Martin, Owen Taylor, Arnaud Charlet und Dom Lachowicz -- LGPL [1]
- libiconv -- Bibliothek für die Umwandlung von Unicode von Bruno Haible -- LGPL [1]
- libpng -- Bibliothek für PNG-Bildsupport von Glenn Randers-Pehrson, Andreas Eric Dilger, Guy Eric Schalnat und Mitarbeitern -- zlib/libpng Lizenz
- libXft -- X Windows Schriftart-Rendering von Keith Packard und Mitarbeitern -- MIT Lizenz
- libXrender -- X Windows Rendering-Extension von Keith Packard und Mitarbeitern -- MIT Lizenz
- pango -- Bibliothek für Textlayout und -Rendering von Owen Taylor und Mitarbeitern -- LGPL [1]
- parsetools -- Python Werkzeuge für Parse-Baum-Umwandlung von John Ehresman -- MIT Lizenz
- py2pdf -- Konvertierungsprogramm von Python Source-Code in PDF-Ausgabe von Dinu Gherman -- MIT Lizenz
- pygtk -- Python-Bindings für GTK von James Henstridge und Mitarbeitern -- LGPL [1]

- pyscintilla2 -- Python-Bindings für gtkscintilla2 von Roberto Cavada und Mitarbeitern -- LGPL [1]
- python -- Die Programmiersprache Python von Guido van Rossum, PythonLabs, und Mitarbeitern -- Python 2.3 Lizenz [3]
- render -- Kopfdateien für X Render-Extension von Keith Packard -- MIT Lizenz
- scintilla -- Source-Code-Editorkomponente von Neil Hodgson und Mitarbeitern -- MIT Lizenz
- zlib -- Bibliothek für Datenkomprimierung von Jean-loup Gailly und Mark Adler -- zlib/libpng Lizenz

**Hinweise**

[1] Die LGPL erfordert, dass wir den Source-Code für alle Bibliotheken, die zu Wing IDE verbunden sind, weiterverteilen. Alle diese Module sind im Internet verfügbar. In einigen Fällen können wir Änderungen vorgenommen haben, die noch nicht in die offiziellen Versionen aufgenommen wurden; wenn Sie eine Kopie unserer Version des Source-Codes für irgendeines dieser Module möchten, senden Sie uns bitte eine E-Mail an info at wingware.com.

[2] Docutils enthält einige Teile, die anderen Lizenzen unterliegen (BSD, Python 2.1, Python 2.2, Python 2.3 und GPL). Siehe die COPYING.txt Datei in der Source-Code-Verteilung für Einzelheiten.

[3] Die Python 2.3 Lizenz ist eine OSI anerkannte Open Source Lizenz. Jede Version von Python unterliegt einer ähnlichen, aber einzigartigen Lizenz; Wing enthält nur Python 2.3.

[4] Nicht OSI anerkannt. Wingware hat vom Autor die ausdrückliche Erlaubnis erhalten, diese Themen weiterzuverteilen.

[5] Nicht OSI anerkannt. Diese GTK-Themen sind weit verteilte Arbeiten, die impliziet in der öffentlichen Domäne sind, aber keine angegebene Lizenz oder Copyright haben. Sie können von Wing IDE entfernt werden, ohne die grundsätzliche Funktionalität des Produktes durch das Entfernen der entsprechend benannten Verzeichnisse aus bin/gtk-bin/share/themes innerhalb der Wing IDE Installation zu ändern.

[6] Nicht OSI anerkannt. Diese Lizenz umfasst jedoch das Recht, sie ohne Beschränkungen zu ändern und zu verwenden.

**Scintilla Copyright**

Die Lizenzbedingungen von Scintalla verlangen, dass wir die folgende Copyright-Anmerkung in dieser Dokumentation einschließen:

```
Copyright 1998-2003 by Neil Hodgson <neilh@scintilla.org>

All Rights Reserved

Permission to use, copy, modify, and distribute this softwa-
re and its
documentation for any purpose and without fee is hereby gran-
ted,
provided that the above copyright notice appear in all co-
pies and that
both that copyright notice and this permission notice appe-
ar in
supporting documentation.

NEIL HODGSON DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS
SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS, IN NO EVENT SHALL NEIL HODGSON BE LIABLE FOR ANY
SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE
OR PERFORMANCE OF THIS SOFTWARE.
```

**Fontconfig Copyright**

Die Lizenzbedingungen von Fontconfig verlangen, dass wir die folgende Copyright-Anmerkung in dieser Dokumentation einschließen:

```
Copyright © 2001,2003 Keith Packard

Permission to use, copy, modify, distribu-
te, and sell this software and its
documentation for any purpose is hereby granted wi-
thout fee, provided that
the above copyright notice appear in all co-
pies and that both that
copyright notice and this permission notice appear in suppor-
ting
documentation, and that the na-
me of Keith Packard not be used in
advertising or publicity pertaining to distributi-
on of the software without
specific, written prior permission.  Keith Packard makes no
```